DOCTOR OF PHILOSOPHY

An integrated approach to deliver OLAP for multidimensional Semantic Web Databases

Matei, Adriana P.

Award date:
2015

Awarding institution:
Coventry University

An integrated approach to deliver OLAP for multidimensional Semantic Web Databases

Adriana P. Matei

PhD

September 2015

An integrated approach to deliver OLAP for multidimensional Semantic Web Databases

Adriana P. Matei

A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philisophy

September 2015

Acknowledgement

Firstly and foremost I would like to express my gratitude towards my director of studies, Professor Kuo-Ming Chao, and my supervisors, Professor Nick Godwin and Dr. Nazaraf Shah. Not only did they support and encourage me throughout this research project, but unknowingly, by doing so, they helped me to mature myself in the person that I am today. Without their support, this work would not have been achieved.

I would like to thank each member of stuff at Faculty of Engineering and Computing at Coventry University which helped me with all the administrative issues I came across the years and let me focus on the work carried on in this research.

Finally I would like to thank my family, especially to Philipp and to all my friends, some of whom were alonside me from the beginning of my journey, some of whom joined along the road, but whom nevertheless understood, encouraged and supported me at different moments during these years.

List of publications

Matei, A. , Chao, K.-M, Godwin, N. (2015) "OLAP for Multidimensional Semantic Web Databases" in *Enabling real-time business intelligence, Lecture Notes in Business Information Processing,* Volume 206, M Castellanos, Springer Berlin Heidelberg, 2015, pp. 81-96

James, A.E., Chao, K-M; Li, W., Matei, A., Nanos, A.G.; Stan, S-D., Figliolini, G., Rea, P., Bouzgarrou, C.B., Bratanov,D., Cooper, J., Wenzel, A., Van Capelle, J., Struckmeier, K. (2013) "An Ecosystem for E-Learning in Mechatronics: The CLEM Project," *Proceedings of IEEE 10th International Conference on e-Business Engineering (ICEBE),* Coventry, United Kingdom, 11-13 September 2013

Matei, A. , Chao, K-M., Godwin, N. (2013) "OLAP for Multidimensional Semantic Web Databases" *Proceedings of Business Intelligence for the Real Time Enterprise (BIRTE) Workshop*, Riva del Garda, Trento, Italy, 26-30 August 2013;

Chao, K-M., Shah, N., Farmer, R., Matei, A. (2012) "Energy Management System for Domestic Electrical Appliances"*, Proceedings of International Journal of Applied Logistics (IJAL),* Volume 3, Issue 4, 48-60

Chao, K-M., Shah, N., Matei, A., Zlamaniec, T., Li,W., Lo, C-C., and Li, Y. (2011) "Intelligent Interactive System for Collaborative Green Computing", *Proceedings of the 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD),* Lausanne, Switzerland, 8-10 June 2011

Shah, N., Chao, K-M., Zlamaniec, T., and Matei, A. (2011) "Ontology for Home Energy Management Domain", *Proceedings of Springer Berlin Heidelberg - International Conference*, DICTAP 2011, Dijon, France, June 21-23, 2011, Part II, 337-347

Chao, K-M., Shah, N., Farmer, R., Matei, A., Chen, D., Schuster-James, H., and Tedd, R. (2010) "A Profile Based Energy Management System for Domestic Electrical Appliances", *Proceedings of the IEEE 7th International Conference on e-Business Engineering (ICEBE),* Shanghai, China, 10-12 November 2010

Abstract

Semantic Webs (SW) and web data have become increasingly important sources to support Business Intelligence (BI), but they are difficult to manage due to the exponential increase in their volumes, inconsistency in semantics and complexity in representations. On-Line Analytical Processing (OLAP) is an important tool in analysing large and complex BI data, but it lacks the capability of processing disperse SW data due to the nature of its design. A new concept with a richer vocabulary than the existing ones for OLAP is needed to model distributed multidimensional semantic web databases.

A new OLAP framework is developed, with multiple layers including additional vocabulary, extended OLAP operators, and usage of SPARQL to model heterogeneous semantic web data, unify multidimensional structures, and provide new enabling functions for interoperability. The framework is presented with examples to demonstrate its capability to unify existing vocabularies with additional vocabulary elements to handle both informational and topological data in Graph OLAP. The vocabularies used in this work are: the RDF Cube Vocabulary (QB) – proposed by the W3C to allow multi-dimensional, mostly statistical, data to be published in RDF; and the QB4OLAP – a QB extension introducing standard OLAP operators. The framework enables the composition of multiple databases (e.g. energy consumptions and property market values etc.) to generate observations through semantic pipe-like operators.

This approach is demonstrated through Use Cases containing highly valuable data collected from a real-life environment. Its usability is proved through the development and usage of semantic pipe-like operators able to deliver OLAP specific functionalities.

To the best of my knowledge there is no available data modelling approach handling both informational and topological Semantic Web data, which is designed either to provide OLAP capabilities over Semantic Web databases or to provide a means to connect such databases for further OLAP analysis.

The thesis proposes that the presented work provides a wider understanding of: ways to access Semantic Web data; ways to build specialised Semantic Web databases, and, how to enrich them with powerful capabilities for further Business Intelligence.

Content

List of Figures

List of Tables

Chapter 1 – Introduction

1.1 Research Context

In today's business, the data (e.g. web data) obtained over the Internet and their semantics can play an important role as resources in enhancing data analysis, when used in combination with internal enterprise business information systems. The Semantic Web (SW) technologies provide the capability of annotating web data with semantics hence generating Semantic Web data.

The information and activities in a typical Business Intelligence (BI) scenario can be modelled by three different layers (Berlanga, et al., 2012): the data source layer, the integration layer and the analysis layer. The combination of Data Warehouses (DWs) and On-Line Analytical Processing (OLAP) covers these layers in order to support BI efficiently. OLAP tools and algorithms have been used successfully in BI to query large multidimensional (MD) databases or DWs for supporting decision making. In the middle layer the multidimensional model is used for normalizing and formatting the data, gathered from other sources, for subsequent analysis. The MD dataset representation is done through the OLAP Cube which is built from the data source using the ETL (extract, transform and load) process.

The evolution of data management on Semantic Webs (SW) has recently showed an increase in the use of the On-Line Analytical Processing (OLAP) approach. Different representations of the OLAP adaptation to a Semantic Web resulted in different structures and vocabularies developments for handling semantic data (Etcheverry & Vaisman, 2012) (Tennison & TSO, 2011) (Chen, Yan, Zhu, Han, & Yu, 2008) (Qu, et al., 2011) (Berlanga, et al., 2012). This triggered a trend in the development of autonomous and usually heterogeneous OLAP databases for SWs. As a consequence data can be found in different OLAP databases on SWs with different query languages to access it; which makes it harder for individual databases to communicate and share with others.

1.2 Research Problem

An increasing number of large repositories containing semantically annotated data are available over the Internet, but summarising the semantic data to support decision making is not a trivial task due to its exponential data growth and complexity issues.

The utilisation of OLAP capability in organising semantic web data into statistical or concise information can increase efficiency in analysis and visualisation. The implementation of OLAP analysis over a semantic web (SW), however, was understood in more than one way and two main types of approach were adopted. Firstly, OLAP is performed after retrieving multidimensional information from a Semantic Web and stored in traditional databases. The second targets the development of OLAP operations directly over Resource Description Framework (RDF) data. As for the first approach, storage of semantic web data in local DWs conflicts with the dynamic nature of web data, as OLAP is designed for static and batch offline processing. In addition, the manually built DWs cannot automatically reflect changes in the sources so that it is hard to maintain the consistency between them.

On the other hand in order to perform OLAP over SW data there are a set of key aspects needed in the modelling process. There is a need for a precise, explicit describing vocabulary in order to represent OLAP data consistently. The key concepts of *dimension* and *measure* need to be introduced to support OLAP operations since these employ *measures* such as AVG, MIN, SUM etc. and *dimension* related actions such as *roll-up*, *dice*, *slice*, and *drill*.

SW data are, however, often published on the web in different cube representations (Etcheverry & Vaisman, 2013) (Tennison & TSO, 2011) for OLAP operations. As a consequence these generated multidimensional semantic web databases become standalone databases, so they only offer limited OLAP capabilities and only work with their own query languages. The information contained in these web databases can be incomplete for complex applications which may require information from multiple databases. Their proprietary specifications do not provide the possibility of direct communication or simple data sharing in order to compose appropriate responses. This is complicated when queries need to be performed over disparate data sources for new multidimensional semantic web databases. The situation could be improved if we could better understand SW data's modelling requirements and model it from an OLAP perspective and so publish it for further aggregations.

After carefully studying the literature available, under the researcher's understanding, the main guiding research question can be defined as:

Q0 – How can we address Semantic Web data in order to provide OLAP capabilities across distributed SWDBs?

The complexity of this research question requires a number of more specific, derived research questions. Answering these questions through this research would provide a complete answer of the main introduced research question. These secondary research questions are:

- ***Q1 – What do we understand by OLAP over SW data?*** By answering this question, the boundary and the context of this research can be defined. The focus of a new modelling vocabulary for OLAP can also be emphasised.

- ***Q2 – Why are the current vocabularies and modelling approaches not suitable to appropriately model SW data for OLAP?*** In order to answer this question we first have to justify why OLAP capabilities are needed over SW data. Then we have to explain why the current vocabularies are not able to deliver an accurate modelling tool. This will lead to the suggestion of a need for a new vocabulary (Integrated Graph OLAP – IGOLAP) to be developed in this research.

- ***Q3 – How can we perform OLAP over the SW's modelled data?*** After we understand what the limitations of performing OLAP on SW data are, we will have the context to develop the necessary OLAP operators, capable of performing OLAP on the data modelled by the introduced IGOLAP vocabulary.

- ***Q4 – How will these new set of operators and vocabulary help improve the communication and OLAP capabilities across shared SWDBs?*** Answering this question will allow the generation of a procedure in applying IGOLAP vocabulary and operators in modelling existing SW data.

1.3 Research Aim and Objectives

The aim of this research is defined, based on the research problem identified in the previous section, as it follows:

The definition and development of both a vocabulary and a set of operators which can be used to model distinct SWDBs and provide them with OLAP capabilities and communication and information sharing facilities

In order to achieve this aim, the following objectives were established:

1. Define the particularities of SW data, describe the differences between informational and topological SW data.
2. Define the OLAP requirements to operate over SW data.
3. Develop a vocabulary to model both informational and topological SW data including OLAP capabilities of dimensions, measures, hierarchies and operators.
4. Assess the IGOLAP Vocabulary
5. Define a set of OLAP operators able to operate on RDF format.
6. Develop the operators.
7. Assess the operators
8. Define how the vocabulary and operators deliver an integrated system for collective querying over multiple multidimensional databases.
9. Demonstrate the benefits of this system.

1.4 Research Approaches

This work is based on a pragmatic approach to research, making use of methods, techniques and procedures from both quantitative and qualitative approaches.

The research methods and techniques selected to fulfil the objectives described in Section 1.3. Research Aim and Objectives are presented in Table 1.1. Their selection is based on the research milestones. First, a good understanding of the research context is established. This lead to identifying the research problem in the above given context. Derived from the research problem was identified the need for SW's data modelling for OLAP capabilities. Then a review of relevant literature was carried out to help develop and assess a specialised modelling vocabulary (IGOLAP) and additional required operators. The next milestone was met when the IGOLAP and the operators were developed and verified on the basis of OLAP modelling and querying over Semantic Web data. And finally, the IGOLAP Vocabulary and introduced operators are proposed and demonstrated.

Research Objectives	Research methods and techniques
1. Defined the particularities of SW data, describe the differences between informational and topological SW data.	Review of relevant literature;
2. Define the OLAP requirements to operate over SW data.	Review of relevant literature;

3. Develop a vocabulary to model both informational and topological SW data including OLAP capabilities of dimensions, measures, hierarchies and operators.	• ***Data collection*** –
	  ○ Data samples collected from relevant literature review (RDF format);
	  ○ Raw data collected from a real-life data energy monitoring data (SQL);
	  ○ Synthetic data collected through online manual data mining;
	• ***Data analysis*** – Analyse the data for retrieving semantic web specific data patterns and properties;

3. Assess the IGOLAP Vocabulary	• ***Data modelling*** – Modelling the sample data from Objective 3 based on the developed vocabulary (IGOLAP) from the same objective;
4. Define a set of OLAP operators over RDF format.	Review of relevant literature (OLAP and RDF querying languages)
5. Develop the operators.	Usage of traditional OLAP operators main characteristics as guideline through the development;
6. Assess the operators.	• Apply the provided operator on data provided by achieving Objective 4; • Evaluate the correctness of the operators' output;
7. Define the integrated system for collective querying over multiple multidimensional databases	Usage of the vocabulary to model multiple multidimensional SW database; Usage of the operators to perform composed queries across these databases;
8. Demonstrate the benefits of the system.	A relevant case study;

Table 1.1 Research plan on achieving the research objectives

1.5 Contribution to Knowledge

This research will mainly contribute to the domain of Semantic Web usage. It intends to provide a new way of modelling SW data for enhancing BI potential in this area through providing an OLAP capability. Satisfying the aim and the objectives from this research will provide three primary contributions to knowledge. Firstly, by using the

introduced vocabulary and operators this research provides an integrated system for collective querying over multiple multidimensional databases. Secondly, this research provides an extended vocabulary for multidimensional data representation. Lastly, it presents an example materialization of a semantic OLAP database capability.

In conclusion this research contributes to the field of data modelling and data integration in the Semantic Web and Linked Data area.

1.6 Thesis Structure

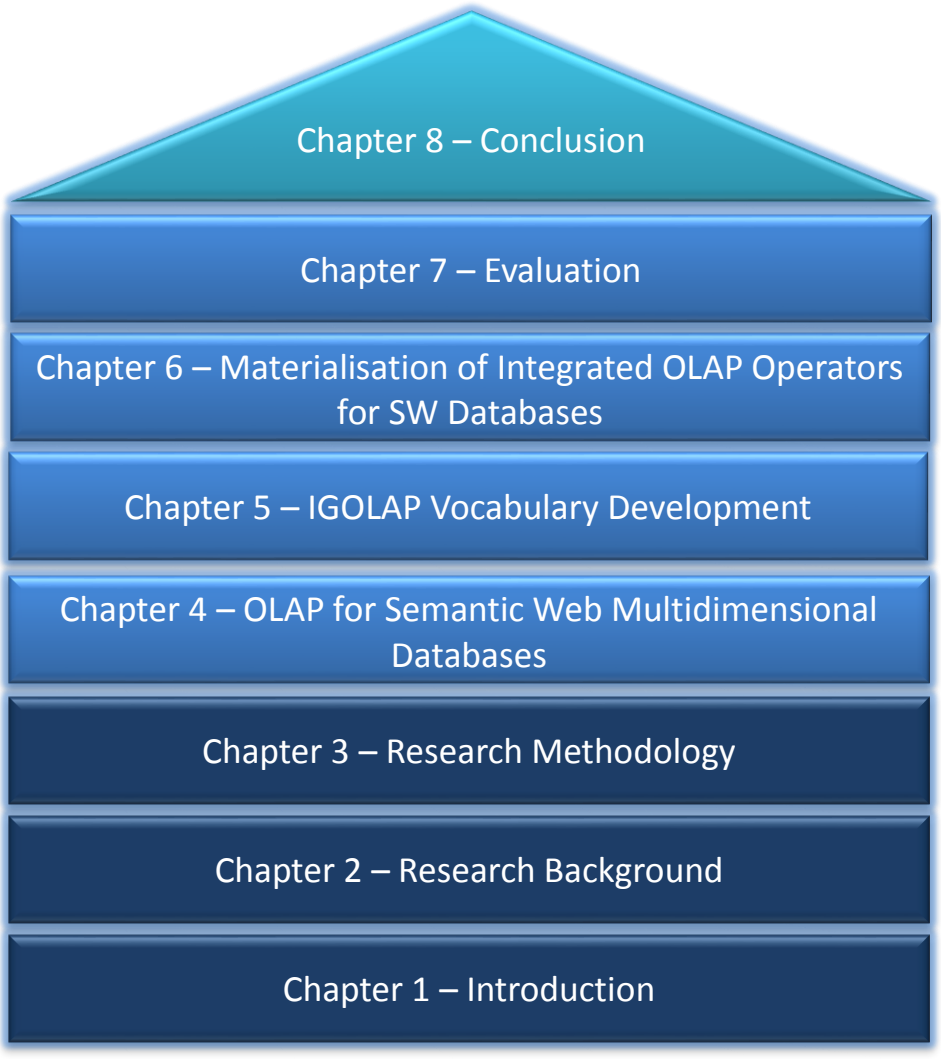


Figure 1.1 Thesis' chapters based structure

This thesis is structured as eight chapters designed to deliver the thesis' content in three stages. Firstly, Chapters 1 to 3 provide the pre-requisites to understand the conducted research work presented. The next four chapters, Chapter 4 to 7, deliver the

thesis' research work. In the final chapter the thesis is concluded and future research visions provided.

The thesis' chapters and above introduced stages are visualized in Figure 1.1 Thesis' chapters based structure.

Furthermore each of these chapters is summarised below:

- ***Chapter 1 – Introduction:***
  In the current chapter the discussion considered the research context and the problem as well as the research approach and the contribution to knowledge. This chapter is concluded with the introduction of the thesis' structure.
- ***Chapter 2 – Research Background:***
  The research context was extracted from an initial literature review, but an additional literature review was essential in answering the research questions. The entire research background is presented in Chapter 2 and focuses on the interactions of two main worlds: on the one hand the Semantic Web world and its tools and technologies and on the other hand the important BI tool – OLAP – and its adoption in the Semantic Web. A critique of the available work and the requirements for solving the research problem are also specified in this chapter.
- ***Chapter 3 – Methodology:***
  The methodological approach used in this research work is described in this chapter. Part of the methodological approach includes the research design and process.
- ***Chapter 4 – OLAP for Semantic Web Multidimensional Databases:***
  In Chapter 4 the overview of the desired output as well as the framework of the research is introduced both graphically as well as being detailed in an explanatory way. This chapter also presents the design and architectural aspects of the work currently delivered.
- ***Chapter 5&6 – IGOLAP Vocabulary Development & Materialisation of Integrated OLAP Operators for Semantic Web Databases:***
  The main two components of the system developed in this research have each their own chapter dedicated. Each chapter introduces the implementation of

the vocabulary and operators based on the methodology previously introduced. The variations of the implementations, the sequence of the iterations over the implementation and special delivered characteristics in each of the final versions are all detailed in these chapters.

- ***Chapter 7 – Evaluation of the Integrated Framework***:
  The entire chapter 6 is dedicated to the evaluation of this research. The evaluation process covers individual components as well as their inter-operability and overall system evaluation.
- ***Chapter 8 – Conclusion and future work***:
  The final chapter of this thesis concludes the findings of this research, it presents how the research aims and objectives were achieved. This chapter discusses over the limitations of this work but also introduces the contributions delivered during the research. After providing the identified future areas of research, this chapter ends this thesis with the concluding remarks.

Chapter 2 – Research Background

2.1 Semantic Web main concepts

The Semantic Web operates with a series of terms and technologies either specifically designed or adapted to represent and process web content in a machine readable way.

One important concept in understanding the Semantic Web is *information*. Information is stored everywhere in one way or another, from big enterprises' databases to web pages as HTML (Hyper Text Markup Language) documents, all can contain data that can be provided, or not, by a database. This means that access to raw data is not necessarily provided. In addition the transition from information intended primarily for human readability into information for computers and machines to process, was viewed as a necessary next step and as a consequence the ***Semantic Web*** was introduced (Berners-Lee, Hendler, & Lasssila, 2001). Overall it represents a group of technologies and ways of making the semantics (meaning) of the information on the Web (World Wide Web) accessible for machines. A description of the Semantic Web was made available by the World Wide Web Consortium (W3C) (W3C Consortium, 2010) and its director, Tim Berners-Lee.

There are many areas in which the semantics can be used. The Semantic Web idea is, ideally, user oriented; trying to understand problems like user's access and the sharing of the data. Most importantly the Semantic Web should facilitate user – machine communication on a level on which applications will understand the meaning of different data and text and be able to make connections between them.

The main idea of the Semantic Web's design was that this will not be only another data model but it will be appropriate to the linking of data of many different models. As a result it will be able to add information relating different databases on the Web, this will lead to the possibility of performing sophisticated operations across them (Berners-Lee T. , 1998). The way in which data was made available on the Web previously to this was mostly as CSV, XML or marked up as HTML tables but in all these cases much of its structure and semantics was lost.

As mentioned in (Beheshti, Benatallah, Motahari-Nezhad, & Allahbakhsh, 2012), the Semantic Web envisages the transition from a Web of documents to a Web of data. In

order to achieve that, the access to data should be made using the Web's architecture and the relationships between data need to be defined and described. On top of that, relationships themselves, between two resources or values, need to be named. Automatic interchange of data relies on the explicit naming and defining of those relationships, which generally is done using the Resource Description Framework (RDF) (Berners-Lee T. , 1998) which has the capability to give a formal definition for that interchange. (Beheshti, Benatallah, Motahari-Nezhad, & Allahbakhsh, 2012)

As publishing and sharing of the data was encouraged, the holders of diverse and heterogeneous datasets needed a common way to integrate data coming from different domains, fields and subfields. In order to achieve this, adopting common conceptualization was considered the first step and these frameworks were referred to as ***ontologies*** (N. Shadbolt, 2006). But this wasn't the only necessary step to be made as the data needed to be published and to address not only document linkage representation but also the documents and the data to be linked in a global information space. In order to provide common guidance for this, a set of best practices was made available under the name of ***Linked Data*** (Linked Data community, n.d.).

2.1.1 Ontologies

In the context of the Semantic Web (and Computer Science in general), an ontology is used to formally describe a domain of knowledge. It describes a set of concepts and the relationship between them within that domain and it opens the possibility of reasoning about the entities from a domain. In other words it provides a mechanism to describe information about the objects and relationships between them in a specific domain, using a defined vocabulary.

The necessity of accessing existing data sources, by more and more organizations, using tools that on top of being flexible should be powerful and efficient was also emphasised in previous research work (Poggi, et al., 2008). Research has been conducted in this area on different aspects such as developing ontology languages (Poggi, et al., 2008), extensions of existing ones (Krötzsch, Maier, Krisnadhi, & Hitzler, 2011) and optimizing ontological queries (Orsi & Andreas, 2011).

As presented in (Poggi, et al., 2008), linking the data source to an ontology through a new ontology language promises to be a step forward towards what linked data tries to deliver. But, in this case, the linkage is done using a mapping language for handling the difference between the elements that represent the data source and the elements of the ontology. These types of initiative do address aspects of representing ontologies. However, when used for accessing a large amount of data they show that such ontological access of that quantity of data would be highly costly from the computational point of view. As a consequence, the presented work, fails to take into consideration general availability aspects of ontologies.

The new direction in Linked Data, Semantic Web (and even in independently derived ontologies) is towards the possibility of publishing, sharing and reusing ontologies. This arises from the desire to enable and facilitate data interoperability. A complex survey (d'Aquin & Noy, 2012) presents the status of the most representative available results regarding the publishing, sharing and accessing of ontologies. As the survey states, it appears that it is more cost-efficient for data providers to reuse available, well-established and tested ontologies than to build from scratch an ontology used solely to describe their data.

In order to be able to reuse ontologies (d'Aquin & Noy, 2012), these need to be published and to be able to be accessed in a specific format. For this purpose, systems for collecting ontologies and making them available have been increasingly developed under different names as: ontology repository; ontology directory; ontology archive, or, ontology library. Regardless the name, they serve the same purpose, to give users the ability to find, reuse and publish ontologies.

The use of standard formats such as RDF makes possible the reuse of data and the linkage of diverse data by guaranteeing the interoperability at the syntactic level.

Further, OWL (Web Ontology Language) has become the commonly adopted language for representing ontologies on the Web. OWL is the latest standard in ontology languages provided by World Wide Web Consortium (W3C) (W3C Semantic Web, 2004) and its full description can be found in W3C Recommendations (W3C Semantic Web, 2004). It is built on top of RDF and RDF Schema (RDFS), which it

extends and it is based on DAML (DARPA Agent Markup Language) and OIL (Ontology Inference Layer). Components of an OWL Ontology are classes, individuals and properties and it is primarily designed to describe and define classes.

OWL has three sublanguages as introduced in (W3C Semantic Web, 2004) and these sublanguages have different expressiveness levels in order to address the diverse requirements of their users. The sublanguages' levels of expressiveness are constrained by their computational completeness and they are presented in three variations as follows: OWL Lite, OWL DL (supporting description logic business segment) and OWL Full. From OWL Lite which offers restricted expressiveness but guaranteed computational completeness to OWL Full in which the expressiveness is maximum but the computational completeness is not guaranteed, they all are extensions of RDF with Lite and DL being extensions of a restricted view of RDF. (W3C Semantic Web, 2004).

Looking at Figure 2.1, the newly adopted OWL 2, which is the successor of OWL, has almost the same structure as its predecessor and the relationship between RDF-Based and Direct Semantics (a direct model-theoretic semantics as described in (Motik, 2010) remains the same. (W3C OWL Working Group , 2012)

As mentioned in (W3C OWL Working Group , 2012), adding different levels of semantics to an ontology, can be done either directly or indirectly. The ontology structures can be amended directly and the resulting semantics are then compatible with the SROIQ description logic, which is an extension of the underlying OWL-DL description logic (Horrocks Ian, 2006). The indirect route is through the mapping of the RDF graphs to the ontology structure where the meaning is directly assigned to the RDF graphs. (W3C OWL Working Group , 2012)

Research has been conducted into different extensions to ontology languages. For example (Krötzsch, Maier, Krisnadhi, & Hitzler, 2011) presents research work conducted into the extension of a descriptive language (DL)-based ontology language. Although only theoretical results have been presented in this case, it shows that this type of extension, for this particular case, nominal schemas extension offers expressivity to incorporate rule-based modelling into ontologies. This has been exemplified by the

integration of rule based languages such as the Semantic Web Rule Language (SWRL) and the Rule Interchange Format (RIF) with OWL 2.

As acknowledged in (Orsi & Andreas, 2011), recent years have shown an increase of Linked Data initiatives and the adoption of Semantic Web tools such as RDF, RDFS and OWL. This has triggered research on techniques for data management of a Semantic Web in order to be able to support large repositories of semantic data. In this context these technologies should address both the querying and the efficient storage of these repositories since the solutions available still often rely on relational database systems to deliver efficiency.

Figure 2.1 The Structure of OWL 2 **(W3C OWL Working Group , 2012)**

Nowadays, although research is still conducted into ontologies, the perspective from which this is addressed has shifted. The quantity of Linked Data research has overtaken that on the improvement of ontology usage. Comparing Linked Data and ontology as standalone concepts and/or approaches, both have benefits and drawbacks relative to each other. Firstly, as discussed in (Studer, Simperl, & Kämpgen, 2011), ontologies are required to have a good balance between effort and the added value that they provide.

On one hand they need to be lightweight in order to be more easily understood and reused. But, on the other hand, the reuse of ontologies is still not fully embraced due to the current designs not providing sufficient benefit for the effort required for their exploitation. Additionally Linked Data follows the current direction for Open Data and has proved that viral growth works well in some respects but they still are heterogeneous, inconsistent and often not trustworthy. Consequently there are benefits that can be foreseen from using ontologies in a Linked Data context and the other way around but these will be discussed after a deeper introduction into Linked Data.

2.1.2 Resource Description Framework (RDF) and its Schema

As introduced in (Klyne & Carroll, 2004): "The Resource Description Framework (RDF) is a framework for representing information in the Web".

RDF's syntax is abstract and reflects a graph-based model. The development and acceptance of RDF was mainly motivated by issues such as: Web metadata; representation; machine-readable information; interoperability between applications, and, automated processing of information available across the Web. One of RDF's expected characteristics is the representation of information in a flexible way with minimum constraints. RDF provides a graph data model which retains data and assertions over resources in a triplet form represented by ***subject– predicate– object*** and it is the main way of representing Linked Data, as presented by the "Linked Data principles" (Berners-Lee T. , 2006). In these triplets the subject, predicates and objects are resources while subjects can be also blank nodes and object literals.

While RDF represents means to deliver statements about resources, the definition of the classes of resources and their properties is done through the use of a set of reserved words – RDF Schema (RDFS).

As mentioned in the previous sections , RDF became the commonest way to represent and address Web ontologies and as well it is a building block of Linking Data. All this makes RDF a very important way of describing data in the Semantic Web approach. In fact RDF was designed to address Semantic Web data representation.

A consequence of this is that most research conducted and addressing Web Ontologies, Linked Data and the Semantic Web are either adding to research into RDF or using RDF for exemplifying the investigation. A collection of this work is presented in (Bizer, Heath, & Berners-Lee, 2009) (d'Aquin & Noy, 2012) (Parundekar, Knoblock, & Ambite, 2010) (Berners-Lee, et al., 2006) (Le, Duan, Kementsietsidis, Li, & Wang, 2011) (Motik, 2010) (Wenzel, 2011) (Kämpgen & Harth, 2011) (Etcheverry & Vaisman, 2012) (Kämpgen, O'Riain, & Harth, 2012).

Figure 2.2 RDF triple and its RDF/XML serialization example

One downside of RDF could be considered to be the lack of RDF browsers that would make the data easy to explore and analyse. Considering the increasing importance of linked RDF data in the Semantic Web and overall in the Web of Data context, making data quickly viewable was acknowledged as a highly desired functionality (Berners-Lee, et al., 2006). Nonetheless, achieving this functionality and generic browsing quality is, as concluded in (Berners-Lee, et al., 2006), highly connected with the expressiveness of the comments included in provided ontologies as these have the purpose of providing applications with the ability to offer views from previously unknown domains.

When retrieving data from RDF the most frequently adopted query language is SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language) (W3C Working Group, 2008) which can query data which is either stored as RDF or viewed as RDF and the result of the queries performed can be either structured as RDF graphs or simple result sets.

2.2 Linked Data

A broad definition of Linked Data is provided by (Linked Data community, n.d.), discussing a variety of interpretations of the term. The introduction emphasise the linkage or connection of related data across the Web through some specific newly designed methods.

A clearer introduction to Linked Data is perhaps done in (Bizer, Heath, & Berners-Lee, 2009), mentioning simply that it refers to using the Web for linking data from different sources and it is has a large level of applicability. It can be used by organizations sharing data from different geographic locations and also for heterogeneous systems within an organization. There is a well-defined set of rules, also known as: "Linked Data principles" and they have the role of guiding the way in which data should be published, in order to become part of a single data space. These principles have been previously formulated in (Berners-Lee T. , 2006) and have since been emphasised by all research work involving Linked Data and therefore are presented below:

1. ***Use Uniform Resource Identifiers (URIs) as names for things***
2. ***Use HyperText Transfer Protocol (HTTP) URI so that people can look up those name***
3. ***When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)***
4. ***Include links to other URIs, so that they can discover more things***

In consequence the two fundamental technologies on which Linked Data relies are URI and HTTP but they are supplemented by RDF which is a technology that is critical for the Web of Data. URI, or even IRI (Internationalized Resource Identifiers (Hyland, Atemezing, & Villazón-Terrazas, 2014)), in addition to RDF, as supported by (Bizer,

Heath, & Berners-Lee, 2009), (Berners-Lee T. , 1998), provides a graph data model which is generic enough to structure and link Linked Data. One of the main reasons for this is the subject – predicate – object triples form in which data is encoded.

One major example of Linked Data principles put into practice is the Linked Open Data project (OPEN GOVERNMENT PARTNERSHIP, 2011) which is a community shared effort supported by W3C Semantic Web Education and Outreach Group (W3C SWEO, n.d.) with the aim of identifying datasets available under open license and to publish them on the Web after a conversion to RDF applying Linked Data principles (Bizer, Heath, & Berners-Lee, 2009)

The openness of governments and public agencies for sharing their data was a consequence of the Open Data initiative. At this time complex data is available under open licences and covers information about geographical locations, scientific publications, books, entertainment areas, bioinformatics, medicine, online communities, statistical data, reviews, companies and many others. More detailed information about publishing linked data is generally available and it is presented as well in (Bizer, Heath, & Berners-Lee, 2009) alongside use and reuse of RDF Vocabularies and URIs. With all this there are three main steps that are involved in the process and they are generally followed. As presented in (Bizer, Heath, & Berners-Lee, 2009) these steps refer to: assigning URIs to entities and the provision of the URIs over the HTTP protocol for dereferencing into RDF representation; linking to other data sources on the Web; and, the provision of metadata about published data. In this context, issues such as usage of terms from well-known RDF vocabularies rather than describing new vocabularies where possible or the common serialization format that it is advised and generally accepted to use, RDF/XML represent an important discussion point, as also identified in (Bizer, Heath, & Berners-Lee, 2009). Additional important discussion topics in the Linked Data context and best practices are aspects on link generation, metadata or publishing tools.

It is worth mentioning that, depending on the publisher, Linked Data can be accompanied by metadata of different types which help the consumers of Linked Data to choose if they want to trust specific data. For example, evidence based on: information about data creation properties; the evidence for RDF links, and, the tracing

of the changes in links, together with technical metadata describing the means of access, and, differentiable URIs, can be used by the consumer. Use of this can help decide if the described data will be: taken as trustworthy; contain the needed information; and, be usefully used. Presented in (Bizer, Heath, & Berners-Lee, 2009) can be found a large variety of publishing tools and all of them support the dereferencing of URIs into RDF descriptions. On top of that some may offer SPARQL querying access to datasets and support the publication of RDF dumps.

Analysis of research on the state of Open Data (Braunschweig, Eberius, Thiele, & Lehner, 2012) points out that a large amount of data, generated by different bodies, was made available for general use at the (OPEN GOVERNMENT PARTNERSHIP, 2011) initiatives which triggered the appearance of various Open Data platforms. This study states that the openness of Open Data platforms depends a lot on the publishing format since many of the available platforms lack of APIs and proper standards. The platforms use proprietary formats or the non-machine-readable publishing of them makes their data not really usable and open. This shows the lack of coordination under which Open Data community is working at the moment. The Linked Data research programme acknowledges the mentioned problems in the Open Data initiative, additionally concluding that sole publishing of the data and offering it to the public for consumption doesn't have many advantages if it is hard to consume. Furthermore it notices that properties such as standardization, machine readability or discoverability need to be taken into account and the data should be published under a common publishing guideline which the Linked Open Data is aiming to achieve.

As mentioned in (Parundekar, Knoblock, & Ambite, 2010), the linkage of data at the ontologies level is another challenge for Linked Data. This work also proposes an approach on linking ontologies and aligning them but problem remains regarding the vocabulary and the designed use and access across multiple standalone linked data resources.

In order to be able to address Linked Data and ontologies, another concept should be understood, this is RDF. As mentioned by (Klyne & Carroll, 2004), Linked Data relies on documents having their data in RDF format. This concept is briefly introduced in the following subsection of this Chapter.

2.2.1 SPARQL – RDF(S) Querying language

SPARQL is the standard query language sustained and continuously improved by the World Wide Web Consortium (W3C) for RDF (W3C Working Group, 2008) (W3C Working Group, 2013). It is a declarative language, which uses Boolean expressions to evaluate filter conditions, including not only the existence but also the non-existence of a filtering pattern. The query evaluation mechanism is based on graph matching, which performs best on a set of pattern selections. A query has a minimum of three clauses that are used, firstly the PREFIX clause provides the namespaces to be used, the SELECT clause provides the format of the result and the WHERE clause encapsulates the constructed pattern to be used. Additional clauses as FROM, CONSTRUCT or DISTINCT are used to provide identification of the namespace used, building a specific graph result or retrieving unique results respectively.

A simple SPARQL query example, based on the RDF dataset example from section 2.1.2, which returns all resources that are of type foaf:Person, the persons URI and their given name, is provided below:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT *
    WHERE {
        ?person rdf:type foaf:Person .
        ?person foaf:givenName ?name .
    }
```

SPARQL also provides aggregated functions that provide summarized information from multiple triplets into one. As some of the same basic aggregations and sorting functions are encountered in SQL, SPARQL also offers: COUNT, SUM, MAX, MIN and AVG functions, as well as HAVING, FILTER, GROUP BY and ORDER BY sorting operations with ASC or DESC sorting option.

Additional inner-outer query combination, or subqueries, are provided through the WHERE clause, while UNION and MINUS operations combine, or compute the difference between, the results of two subqueries.

With this set of functions, SPARQL provides the basis to develop OLAP aggregations such as roll-up, drill-down, slice or dice when the data is modelled to support these types of complex queries.

2.3 OLAP in conjunction with the Semantic Web

In order to be able to apply OLAP principles over Semantic Web data, the fundamentals of OLAP needs to be understood and the challenges of applying them over Semantic Web databases need to be identified. In this section the concepts and definitions used to guide this work are presented. The challenges identified in this sections are also based on the provided understanding of OLAP.

2.3.1 OLAP Fundamentals

In the context of business intelligence, a set of processes, architectures, systems and technologies are used to process raw data into information. This information is then used in the decision making processes. For a long time, one of the most used mechanisms for structuring, aggregating, analysing, querying and exploiting the data in this context has been **online analytical processing** (OLAP). OLAP terminology was coined by (Codd, 1993) as describing the set of requirements needed to summarise, consolidate, synthesize and view data accordingly to multiple dimensions.

Data warehousing is primarily used in organisational decision making as a "subject-oriented, integrated, time varying, non-volatile collection of data" (Inmon, 2005). That assumes that this is the place where various, historical, collections of data are integrated for further analysis. As presented in (Chaudhuri & Dayal, 1997) (Agrawal, Gupta, & Sarawagi, 1997), in order to be able to deliver complex analysis and visualisation, this data is generally modelled in a multidimensional way. Querying this data and the response time of ad hoc, complex queries across spread data requiring multiple joins and aggregates is more important than transactional throughput. As such, the OLAP

approach is the appropiate one as it delivers the possibility to perform OLAP operations as *rollup, drill-down, slice and dice* or *pivot*.

The primary function of data warehouses is to provide data from different sources, cleansed and customised (Chaudhuri & Dayal, 1997), ready to be filtered, aggregated and then even stored in smaller data stores or data marts. The approach to perform these operations is provided through OLAP applications (Vassiliadis, 1998). But in order to provide these functionalities, the data has to be stored in a multidimensional way. This multidimensional way is represented by a Cube concept, where a Cube define a group of data cells arranged by the dimensions of the data (OLAP Council, 1997). In view of this definition, a dimension is defined as "a structural attribute of a cube that is a list of members, all of which are of a similar type in the user's perception of the data" (OLAP Council, 1997). Furthermore the aggregated data can be viewed inside a dimension based on different levels of details. These levels denote the hierarchical structure of a dimension based on levels. Furthermore the real measured values are represented by measures, variables, facts or metrics and they can also be referred to in these terms (Vassiliadis, 1998).

As presented in the previous paragraph, the main two aspects to be considered in the OLAP systems are: dimensions and aggregations. Each dimension represents a perspective over the data and each of these dimensions or perspectives are complementary to each other. Aggregation levels are the other important aspect of OLAP systems and they are a defining concept of OLAP. Based mainly on these two aspects we can consider different operations within OLAP. On one hand the multidimensional aspect supports “slicing” and “dicing” operations along multiple dimensions. They also support “pivot” or “cross table” operations where the direction of the analysis can be changed. On the other hand aggregations offer “drill down” and “roll up” operations for a view including more or less details from the data as well as saving analysis time through the pre-calculated aggregations.

As mentioned earlier, there are different representations of OLAP systems, depending of the purpose that they need to serve. Basic representation models are the Relational OLAP (ROLAP) and Multidimensional OLAP (MOLAP) as well as the Hybrid OLAP (HOLAP). The main difference between MOLAP and ROLAP

architectures, as presented in (Chaudhuri & Dayal, 1997) and (Vassiliadis & Sellis, 1999) is where the multidimensionality of data is represented. In MOLAP architecture it is provided a direct multidimensional view of the data through the usage of a multidimensional storage engine. On the other hand, in a ROLAP architecture a multidimensional interface to relational data is built. In this approach, usually, a specialised middleware extends the traditional relational servers to support multidimensional OLAP queries. There are identified pros and cons of this models, but the benchmarks comparisons and evaluations of these is out of scope for this research work.

One important aspect in regards to OLAP Systems is that alongside the benefits that they offer, OLAP Systems have problems in modelling issues like: traceability, shortest paths or social networks. These are, however, handled by graph structures and their properties. The latter can address real-life application queries, which otherwise are not properly supported by OLAP systems.

As per defined in (OLAP Council, 1997) the OLAP operators definitions are extracted and then summarised below:

- A ***roll-up*** operation assumes a data summarization inside a given cube alongside a given dimension such as a given Cube C, a dimension $D \in C$ and a dimension level $lu \in D$, the Roll-up(C,D, lu) will return a new cube C' where measures are aggregated along D up to the level lu as presented in below:

Figure 2.3 Traditional roll-up OLAP operator (Tutorial Point, n.d.)

- In the ***dice*** operation a new cube C' is generated from a given cube C and a set of constraints along its dimensions. The emerging cube has the same schema as the initial cube C and the instances in C' are also instances of C.

  Through performing a dice operation a smaller cube is extracted from the given cube by removing the other members from the dimension, without changing the given level of the dimension.

Figure 2.4 Traditional dice OLAP Operator (Tutorial Point, n.d.)

- ***Slice*** operation receives a cube C, and a dimension D ∈ C and returns a sub cube C', with the same schema except the dimension D.

  This operation it is also referred to as "two dimensional page" – selection or "page display" (OLAP Council, 1997). Furthermore, the generated data from a slice operation defines a "spreadsheet" – like data view.

- ***Drill-down*** is considered to be the reverse of roll-up operation and assumes the disaggregation on a previously stored aggregation, navigating through the hierarchical path of a given dimension.

Figure 2.6 Traditional drill-down OLAP operator (Tutorial Point, n.d.)

These are the definitions used in these work as reference for extrapolating the standard operators to the needs of the graph structures of the Semantic Web.

2.3.2 Difficulty in providing OLAP systems over Semantic Web Data

On-Line Analytical Processing (OLAP) has been undeniably proven a successful approach to analysing large sets of data (Berlanga, et al., 2012). Furthermore OLAP is an approach that can be built on top of different database models and respond to multi-dimensional queries as long as they fall under some evaluation criteria regarding, but not limited to, multidimensionality, accessibility, transparency, dimensions and aggregation levels.

The characteristics and the relationship of the data of the Semantic Web can be divided in two categories: informational (dimensions are coming from node attributes) and topological (when dimensions are coming from node and edge attributes). Recently, a considerable stream of works (Etcheverry & Vaisman, 2012) (Tennison & TSO, 2011) (Chen, Yan, Zhu, Han, & Yu, 2008) (Qu, et al., 2011) (Berlanga, et al., 2012) (Chen, Yan, Zhu, Han, & Yu, 2009) (Zhao, Li, Xin, & Han, 2011) (Etcheverry & Vaisman, 2012) was directed towards online analytical processing on informational network and mostly focusing on the Semantic Web data. (Chen, Yan, Zhu, Han, & Yu, 2008) (Chen, Yan, Zhu, Han, & Yu, 2009) (Zhao, Li, Xin, & Han, 2011) and (Qu, et al., 2011) take the first steps towards introducing graphs in a multidimensional and level context by proposing conceptual frameworks for graph data cubes and a data warehousing model able to support graph OLAP queries. They both consider attribute aggregations and structure summarization, where the authors in (Chen, Yan, Zhu, Han, & Yu, 2008) classify their framework into topological and informational OLAP based on the dimension. They proposed different aggregation functions to build summarisations for each dimension and these cannot be mutually applied.

(Kämpgen & Harth, 2011) introduce linked data transformations for OLAP analysis and in (Kämpgen, O'Riain, & Harth, 2012) they attempt to map statistical Linked Data to an OLAP to conform to the RDF Data Cube Vocabulary (Tennison & TSO, 2011) but they did not provide sufficient semantics required for the topological elements to

build parts of the multiple dimensions. (Etcheverry & Vaisman, 2012) introduces Open Cubes which focus on the publication of multidimensional cubes on the Semantic Web and they found the limitation of the RDF Data Cube (Tennison & TSO, 2011) which can only address statistical data. Their work revolves around the informational OLAP's aggregations. As such they revise the RDF Data Cube (DC) by extending its capabilities in order to support multidimensional levels, to build hierarchies and to implement additional OLAP operators. The DC Vocabulary describes only the Slice operator.

(Beheshti, Benatallah, Motahari-Nezhad, & Allahbakhsh, 2012) continue the work from (Chen, Yan, Zhu, Han, & Yu, 2008) (Chen, Yan, Zhu, Han, & Yu, 2009) (Zhao, Li, Xin, & Han, 2011) (Qu, et al., 2011) and offers a graph data model for OLAP informational networks. The approach supports the description of entities and relationships between them and provides topological aggregations. They use three levels of partitioning conditions to implement their proposed model as well as an adapted query language extended from SPARQL in order to support necessary n-dimensional computations. The aforementioned works do not only show the diversity of the approaches towards online analytical processing of Semantic Web but also the rapid change in the research direction.

Table 2.1 Comparing QB and QB4OLAP Vocabularies

The RDF Data Cube Vocabulary (QB) (Tennison & TSO, 2011) focuses on the adherence to Linked Data principles while publishing statistical data and metadata using RDF. QB4OLAP (Etcheverry & Vaisman, 2012) introduces an extended vocabulary for QB in order to support OLAP operators directly over RDF representations. As seen in Table 2.1 Comparing QB and QB4OLAP Vocabularies QB4OLAP introduces levels, members and aggregated functions in order to represent an OLAP dimension structure which is not offered by QB vocabulary. With all these, QB4OLAP, however, does not support a vocabulary to model online analytical processing on graphs introduced by (Chen, Yan, Zhu, Han, & Yu, 2008). (Zhao, Li, Xin, & Han, 2011) introduced a data warehousing model that supports OLAP queries on graph and the Graph Cube. None of these (Linked Data community, n.d.) provides a semantic-driven framework considering both informational and topological dimensions of graphs. (Beheshti, Benatallah, Motahari-Nezhad, & Allahbakhsh, 2012) concentrate their approach on topological graphs without considering informational graphs. This is an important factor as semantic data is usually found in a mix of topological and informational graphs. Furthermore, in order to address topological dimensions constraints for OLAP, they use partitioning and an adapted SPARQL query to operate over the data. This approach hinders the published datasets being reused or being queried by applications and users against other datasets offering automated OLAP observations.

In order to reuse and extend existing implementations while extending OLAP capabilities to both topological and informational dimensions, we used the vocabulary in QB and QB4OLAP as a basis to form a new vocabulary. Furthermore we introduced new elements and relationships able to model the topological OLAP. By describing topological and informational elements in the same vocabulary and identifying the relationships between entities we enable OLAP to operate over both aspects.

Research on bringing the pipe concept to the Semantic Web was introduced by (Morbidoni, Polleres, Tummarello, & Le Phuoc, 2007), where their focus was to build RDF-mashups by fetching RDF models on the Web and producing an accessible output. While the Semantic Pipes operators can access different RDF graphs and produce outputs to be consumed by other pipes, they do not offer means to access summary data or support OLAP operations.

2.4 Summary

The brief introduction of the up-to-date research in the Semantic Web's data management shows that a new model is required to answer computational intensive semantically queries but no existing OLAP system is capable of accessing, retrieving, and reusing semantic OLAP databases efficiently. In order to address this challenge we introduce a new model which can interpret a query based on the OLAP concept. The model offers standard OLAP functionalities with a built-in Pipe concept by extending existing OLAP systems with observations generated from individual RDF graphs or other SW OLAP. This new model is equipped with facilities for composing multiple queries to operate on multiple OLAP databases. It also provides an extended vocabulary for modelling semantic data for OLAP operations.

The challenges presented in this chapter show the need of a specifically designed research methology to address the research problem introduced in this work. This methodology combines diferent methods and these are introduced in the following chapter.

Chapter 3 – Methodology

As presented in Chapter 1.4 – Research Approaches, the research approach used in this work combines methods from both qualitative and quantitative research methodologies. Supporting literature for combining research methodologies (Cook & Reichardt, 1979) (Jick, 1979) (Kaplan & Duchon, 1988) (Greene & Caracelli, 1997) presents the advantages of such an approach, but an important factor is still represented by the selection of appropriate methods and techniques of each particular methodology.

The following subsections introduce the research process and the research design. During the research process, discussed in the following section, a set of methods and techniques were used to achieve the necessary steps. In Figure 3.1 it can be seen how these methods, specific to each methodology, are used as building blocks in the final methodology of this research. The research approach used in this work is presented in details in the next subsections by describing the research process and design.

3.1 Research Process

Following the guidelines introduced by (Kothari, 2004), the activities that describe the research process can be summarized as follows:

- *Formulating the research problem*
- *Extensive literature survey*
- *Development of working hypotheses*
- *Preparing the research design*
- *Determining sample design*
- *Collecting the data*
- *Execution of the project*
- *Analysis of data*
- *Hypothesis-testing*
- *Generalisations and interpretation*
- *Preparation of the report of the thesis*

In this research work these activities were performed in parallel at different stages and their sequence is summarized in Figure 3.1. Beside the *Generalisations and interpretation* and the *Preparation of the report of the thesis* activities, all the other activities are repeated in an iterative manner during the research progress. The reason for this iterative repetition of the activities is due to the research design, where different

methodological technologies and methods describing the same research activity are designed to be used in different stages of this research work.

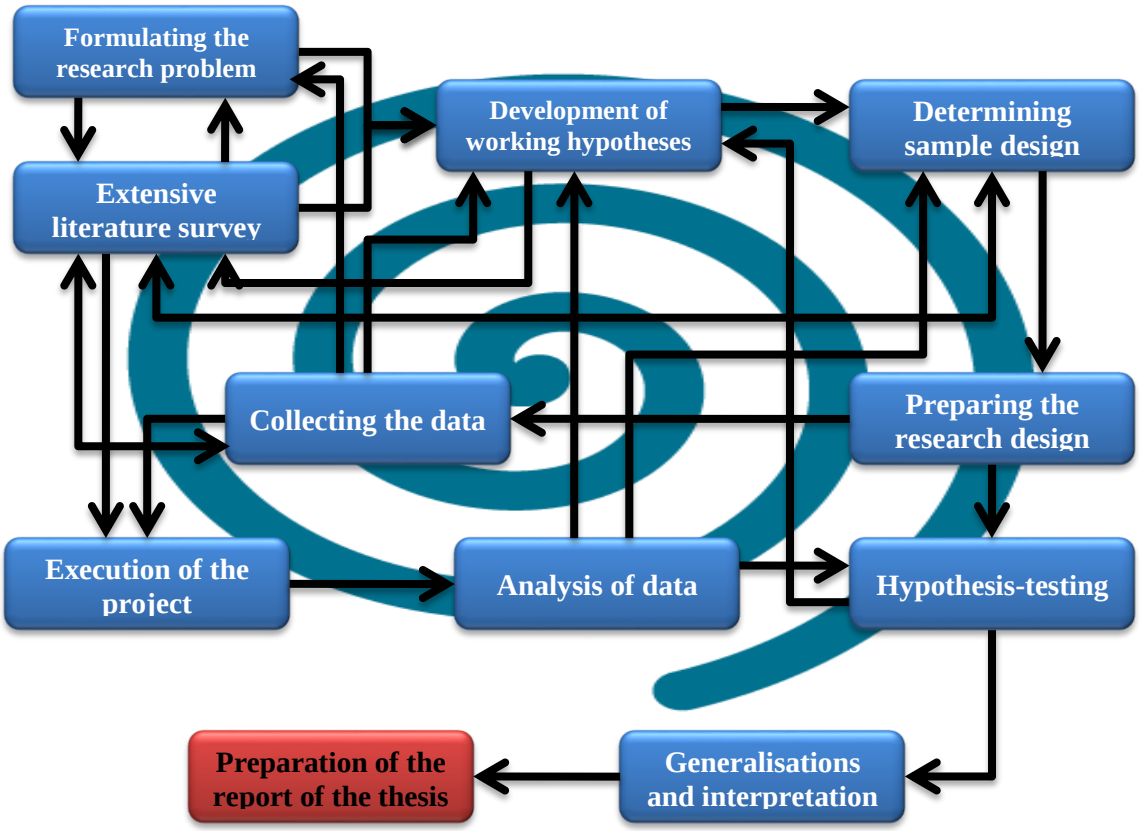


Figure 3.1 Activities inside the research process

Furthermore the details of these activities can be seen, not only through the chapters of this thesis, but also in the following subsection –3.2 Research Design– which additionally introduces the methods and techniques used in this research.

3.2 Research Design

The conducted research work was initiated in the context of the DEHEMS (EU 7th Framework Programme, 2008) project where one of the targets was understanding the semantics behind the energy consumption of household across Europe.

The design of the overall sequence of steps followed in this research are presented in this section, together with the methods used in each of them. Although the research was initiated as part of the above presented European Project, the methodology was designed based on the encountered problem that needed to be addressed, independent of

the carried out work in the project. Furthermore the research methodology designed for this research work consists of 6 Steps and the European Project provides the initial data collected in Step 1. The entire steps and the methods used are presented below:

Step 1: As mentioned by (Kaplan & Maxwell, 2005) qualitative research methodologies' methods can be successfully applied in relation to Computer Information Systems. The methods used in this step from the qualitative research methodologies are:

- *Data collection*
- *Data observation*
- *Data analysis*

These methods were used with the goal of constraining the research to the particular research problem and to be able to derive the hypotheses and predictions based on the observations of the data during the data collection phase and literature review. The Semantic Web domain is rich in semantics, with a vast area of application, and needed an in depth understanding of the context, the particularities of the data and the research problem, in order to construct the research hypotheses.

The initial Data Collection method is reflecting the raw data (time series) collection, in the context of the mentioned project, from around 250 houses across United Kingdom and Bulgaria from the first cycle of data available. The sample dataset, although very large, was observed to be within a specific domain of interest: energy consumption of home appliances. The existing and ongoing development of an ontology describing the entities producing the raw data was already limiting the variation of the starting sample datasets and triggered the context of a literature review on Semantic Web Technologies. Using the Data observation method, the characteristics and particularities of the data were identified. As shown by (Patton, 1990), the context of the research (described in section 1 of Chapter 1) together with the sample data required a purposeful approach to research. The starting sample data comes from using "Maximum Variation sampling" with the existing characteristics of the data, which were outlined through the Data Analysis Method from the qualitative research approach.

The outcome of this step was the initial hypothesis formulated as:

Hypothesis 1: "OLAP operations over Semantic Web data structures require a specific vocabulary for modelling the data and adapted operators, in order to gain the benefits of performing OLAP operations over semantic web databases"

Figure 3.2 Research methodology overview

Based on the observations from Step 1 a specific methodology was designed as presented in Figure 3.2 Research methodology overview, in which the next Steps of the research were designed. By using these methods described in Step 1, it was identified that a system needs to be build and to set the context in which this system needed to be implemented and developed, as well as gain a better understanding to be used in explaining the events, processes and outcomes of particular cases and further to extrapolate from them through ***quantitative research methods***.

It was identified that in the next steps it is needed to extract the research questions through qualitative methods and quantitative research methods need to be used to investigate these questions. Combining a quasi-experimental quantitative research design (Gall, Borg, & Gall, 1996) with models specific to scientific research in computer sciences (Elio, et al., 2011), quantitative data needed also to be introduced in order to be able to test the generated hypotheses but also to remain objective in the design and implementation of the needed solution.

Starting with Step 2 the following identified methods were introduced:

- *Model method* to define the abstract model of the entire system (containing a vocabulary and operators' general overview and connection)
- *Build method* to build the components of the abstract model. Techniques such as system design, components reuse, adequate programming languages and continuous testing were used.
- *Experimental method* is used on two levels, on one hand to identify the questions that the system is expected to answer in the evaluation phase and on the other hand to help answering this during evaluation.

The introduced methods and techniques alongside intense literature review are used in parallel in different stages of the research and characterize the research design.

The research design was developed guided by the research process' activities and the aims and objectives of this research outlined by the outcome work from Step 1. Based on the provided hypothesis it was already known that the research need to focus on two main points: a vocabulary and a set of OLAP operators for Semantic Web data. This influenced the design of the research by defining the context of the research. Starting with this point, the design was independent of the presented European Project, and was designed as following:

Step 2:

- The hypothesis formulated in Step 1 needs to be analysed and improved;
- In order to improve the hypothesis, further data analysis and data observation needs to be carried on;

- The niche literature review should provide the required knowledge of the state of the art technologies and research approaches available in the defined domain;

- Based on a model method, the characteristics of a system able to support the improved hypothesis in relation with the findings from the niche literature review will be defined;

Step 3:

- New relevant data sets from the Semantic Web needs to be identified:

- Based on the output model from Step 2 and further performed niche literature review, the build method needs to be used in order to identify if any available and reusable components exist, which programming languages are needed, design the system architecture and define the testing methods for the identified components;

  In this Step will also be identified if other ontologies or vocabularies exist on the defined domain of knowledge and if not which methodology should be applied to design such a vocabulary.

- Using the experimental method, the set of questions that need to be answered in the evaluation phase are identified.

Step 4:

The two components of the system need to be implemented and continuously adjusted using three methods:

- Build method for implementing and adjusting the components through continuous testing

- Qualitative method for data observation over the validity of the vocabulary and the results of the operators

- Experimental method in order to continuously evaluate the vocabulary against the set of questions and the operators against the identified requirements

Step 5:

- The two components are part of a modelled system, as such the system needs now to be evaluated as a whole against the predefined evaluation questions and the identified evaluation benchmarks;

- The outcome of the previous steps as well as of the final evaluation need to be analysed and interpreted, as such the contextual and narrative analysis approach (Kaplan & Maxwell, 2005) needs to be used at this step.

- The experiment is designed to evaluate the findings and the components that are built in the previous steps. The evaluation design and outcomes have a dedicated chapter, Chapter 7 – Evaluation, and in section Evaluation Design and Process it is presented the methodology used in designing the evaluation as presented in Table 7.1 Key four evaluation requirements and their addressability. The evaluation design considers the evaluation requirements introduced for domain related benchmarks (Grey, 1993)

Step 6:

- Consist of the actual preparation of the performed work and results in a written form.

Since different technologies and methods were used as part of the same process activity, an activities-methodology's methods mapping matrix is presented in Table 3.1. As presented in this table, each activity is composed from one or more methods or techniques and can be repeated across different steps until the final outcome is achieved.

Table 3.1 Research process and design mapping

3.3 Steps and methods of the research methodology

It is presented in the previous section how the methodology of this research work is designed and each needed method identified for the presented steps. In the following paragraphs it introduced the outcomes of each step and how this influenced how the research work it is performed in the next steps. Additionally, the identified evaluation criteria are presented accordingly to each step.

Step 1 outcomes:

- it was identified that the relevant domain of the Semantic Web research is the usage of OLAP aggregation on semantic data.

- The initial hypothesis is formulated: ***Hypothesis 1: "OLAP operations over Semantic Web data structures requires a specific vocabulary for modelling the data and adapted operators, in order to gain the benefits of performing OLAP operations over semantic web databases"***

- Households profiles and energy consumption data is the first collected data

Repercussions on next steps: It identifies the need of a vocabulary and a set of operators which influences the step 4 of the research design

Step 2 outcomes:

- The hypothesis previously formulated can be adjusted based on additional literature review is conducted alongside data observation and further analysis. ***Hypothesis 2: "OLAP operations over multidimensional Semantic Web databases require a specialized vocabulary for modelling both topological and informational data and adapted operators, none of which is currently fully available"***

- An initial model is identified. This is presented in detail in Chapter 4 in Conceptual framework and Architecture Overview sections. Although in Chapter 4 it is presented the final model, this was refined throughout the other steps of the conducted research.

- The research context and data observation is provided by the two use cases presented in detail in Chapter 4 – Architectural Overview and Case Studies

Repercussions on next steps: These outcomes however, have no impact on the next steps presented in the research design.

Step 3 outcomes:

- A set of queries that need to be answered in the evaluation phase are now available as:

Query 1: Monthly energy consumption of a dataset (cube) containing daily

consumption measurements.
In this query, the result is the monthly energy consumption is obtained by aggregating the values of "Day" members in that month. The measure remains the same, but the value of the new observation's measures are the SUM of the values of the daily measures. If there are other dimensions in conjunction with which the measure is defined, this will remain on the same level. For example, when both "location" and "time" dimensions are in the observations, the "city" level in "location" dimension will not be removed or altered when aggregating on a "time" dimension level.
Query 2: Average daily consumption based on households' income ranges.
This query needs to navigate through a given cube from the "Household" to the "Income" dimension. The outcome of this query lists all the Income members, and each member's average energy consumption is calculated based on the households linked to it. All other dimensions remain on the same level of detail, as this query shares the same principle as Query 1.
Query 3: Count the number of days in a month that have a recorded energy consumption.
Given a set of daily consumption observation, per city, per household, this query will aggregate in the "Time" dimension from "Day" level to "Month" level. The measure's value is obtained by counting the number of days for each household and city that has a value.
Query 4: Retrieve the days that had a recorded energy consumption, and the values, used to aggregate the monthly energy consumption.
This query is the reverse operation of Query 1. The daily energy consumption is retrieved only for the months specified in the input dataset.
Query 5: From the daily average energy consumption per Income, retrieve the households and their daily energy consumption for the recorded Income ranges.
The target of the query is to retrieve all the Household members in an Income range from the input dataset. This is an aggregation operation from one topological dimension to another topological dimension.
Query 6: Extract the February month from the monthly energy consumption of the

given dataset (cube).
When given a dataset with energy consumption on the "Month" level in the "Time" dimension, this query should return only the observations over the "February" member. If there are multiple observations containing this member in conjunction with other dimension's members, all of these observations should be returned.
Query 7: Extract a specific household's monthly consumption from a give dataset.
This query is the same as Query 6 with the difference that the member in this case is a specific household, belonging directly to a topological dimension.
Query 8: Retrieve all consumption observations for Birmingham city, for the months of February and March.
The query will retrieve only those observations containing specific members. Only the required members and their measure will be displayed. The above mentioned members belong to all levels in informational dimensions.
Query 9: Retrieve all monthly consumption observations for a specific household.
This query would return all the months containing a measure and its value, for a given household member. This member belongs to a topological dimension. Only the specified member and its energy consumption measurements would be retrieved.
Query 10: Retrieve all consumption observations for Birmingham city, for Months February and March for two specific household.
This query requires an aggregation along specified members from both informational and topological dimensions. As in the previous two queries, only the members specified and their measure will be provided.

Table 3.2 Queries' detailed description and expected outcome

- A set of requirements for the vocabulary and the operators is formulated:
    - Vocabulary should model both topological dimensions in informational data, covering all Semantic Web data structure, in regards to dimensions, levels and measures representation, which are required in order to perform OLAP analysis;
    - The operators should be able to provide materialised and visualisation outputs in regards to all 4 operations: roll-up, slice, dice and drill-

down. The operators should be able to perform over informational and topological data;

- Additionally it is identified in this step that there are ontologies available that could be considered as vocabulary to model Semantic Web data for OLAP.

Repercussions on next steps: In Step 4 the identified ontologies need to be evaluated and the need of considering the ontology design methodology in the build method applied.

Step 4 outcomes:

- The operators' structure defined as presented in Architecture of Federated OLAP Operators (F_Operators), where the federation approach is designed as semantic federation, by modelling the data using the same vocabulary.

- During the evaluation of the identified ontologies, came out that a viable candidate as a basis for the needed vocabulary was available. Partially applying the ontology design methodology as presented in (Uschold & Gruninger, 1996) and (Noy & Mcguinness, 2001).

  Since the needed vocabulary is based on a knowledge-level modelling framework, addressing the building of an OWL ontology adhering to the Semantic Web standards, the methodology chosen to build and evaluate the vocabulary built was the one also sustained by Protégé (Noy & Mcguinness, 2001). Details on standards in ontologies for the Semantic Web are presented in Chapter 2 – Section 2.1.1 Ontologies.

  This methodology and ontology development guideline was applied to evaluate the existing vocabularies, to extend them with the required knowledge representation and evaluate the final vocabulary. As presented in Chapter 5 – IGOLAP Vocabulary Development.

  The methodology steps applied, as described in detail in Chapter 5 – IGOLAP Vocabulary Development, are:

1. Definition of the domain and scope of the vocabulary/ontology (Introduction)
2. Consideration of reusing existing ontologies (Identification of a base vocabulary)
3. Enumeration of important terms in the ontology (Identification of the limitations of the base vocabulary)
4. Definition of classes and class hierarchy (IGOLAP Vocabulary and possible OLAP Operations)
5. Definition of properties of classes (IGOLAP Vocabulary and possible OLAP Operations)
6. Definition of the classes types (IGOLAP Vocabulary and possible OLAP Operations)

The required ontology should support the modelling of all semantic web graph structure for the OLAP analysis.

Repercussions on next steps: No impact on Step 5, which was performed fully as presented in Research Design section.

Step 5 outcomes:

- The full evaluation as presented in Chapter 7 – Evaluation
- The contextualisation and interpretation of the conducted work and evaluation as presented in Chapter 8 – Conclusion and future work

Step 6 outcomes:

- The written outcome in the structure to the presented thesis.

3.4 Summary

The methodological design used in this research was provided in this chapter. This was achieved by introducing the *Research Process* and the *Research Design* used in this research work. The Research Process provides a well-defined set of activities that were followed in this work, while the Research Design provides specific methods and techniques from different research methodologies, which were combined over a set of six steps.

The content of all these steps is discussed in more details across the chapters of this thesis. The entire literature review was concentrated in Chapter 2, the identification of the research questions and the evaluation questions are presented in Chapter 1,

subsections 1.2 and subsection 1.3 as well as in the evaluation chapter –Chapter 7. The model of the system is presented in the next chapter – Chapter 4, while the entire Step 4 from the research design can be seen in Chapter 5 and 6. Chapter 7 and 8 provides the content identified in Step 5 of the research design, while Step 6 represents this entire thesis delivery.

Chapter 4 – Architectural Overview and Case Studies

4.1 Introduction

In this research work two case studies were analysed:

- ***Case Study I: The energy management for domestic electrical appliances***
- ***Case Study II: The household's energy consumption profile with market value composition***

The first case study provided the initial research context and data required to extract the first characteristics and observations in order create the prototype of the system delivered in this research work. This is derived from the time series structure of the data collected from the DEHEMS project. Additional to this data was also collected contextual information, providing household's semantic information, as for example incoms, type of appliances used (make and models), number of inhabitants and ages, etc. The second case study is an extension of the first one, providing a better understanding of the applicability of the research work on generalised real world use case scenarios. This case, is relevant to the contextual information of the households, like address, general state of the propriety based on the value on the market etc. Furthermore, the second case study validated the hypothesis made on the first case study and provided additional information in order to finalize and validate the work in this research.

4.1.1 Case Study I: Energy Management for Domestic Electrical Appliances

As part of the Digital Environment Home Management System (DEHEMS) Project (EU 7th Framework Programme, 2008) it was a challenge to provide the users with effective and focused advice on their energy consumption in order to improve their consumption. In the collection of work related to this project (Chao, et al., 2010) (Shah, Chao, Zlamaniec, & Matei, 2011) (Chao, et al., 2011) (Chao, Shah, Farmer, & Matei, 2012) a number of issues are discussed. The household and energy consumption profiles of household appliances were collected. The large volume of generated data from energy consumption' monitoring sensors were summarised into meaningful information for an intelligent system to reason with. Part of this work materialised into an approach to build an ontology for the home energy management domain. This ontology,

compatible with Suggested Upper Merged Ontology (ArticulateSoftware, n.d.) (Shah, Chao, Zlamaniec, & Matei, 2011), identified and classified the attributes that contribute to the overall appliances' energy consumption.

Figure 4.1 Identified topological and informational dimensions in collected data

As it is presented in Figure 4.1, in the process of semantic modelling of the data identified the need for representing both the topological and the informational dimensions of the data. Additionally these dimensions are not independent of each other but highly interlinked. The two types of dimensions, as well as the full description of their attributes are introduced in the following sections of this chapter and detailed in Chapter 5 – IGOLAP Vocabulary Development.

4.1.2 Case Study II: Household's energy consumption profile with market value composition

The second case study looks into the situations in which it is beneficial and desirable that multiple databases can be accessed simultaneously, by complex queries, in order to provide adequate answers. In this subsection such an example is introduced in which consideration is given to two different databases with complementary information, which, if they are able to communicate, can provide the consumers with complete and valuable information.

One large Semantic Web OLAP database, DB1, contains detailed energy consumption information of households from different countries as well as household attributes such as household income, accommodation size/layout (number of rooms), number of inhabitants, appliances and so on. A separate Semantic OLAP database, DB2, contains information about the historical value on the market of specific properties and their layouts.

Energy consumption for households can be viewed in conjunction with different factors such as: number of inhabitants; household income; house size, or, house value, in order to analyse correlations with energy consumption. The house energy efficiency profile can be a factor in the house acquisition or renting process, so it is desirable to have access to composed information. For example, a natural language version of a query relating to average energy consumption for houses within a selling price range and having a set of other characteristics may be issued by the users as follows:

Find the yearly average electricity consumption in Bristol for households with detached – terrace houses, with 4 occupants and an average selling price between 350000 and 450000 £ (meaning both actual and historical)

This new aggregation can be materialized and stored as a new observation in the queried OLAP or in an independent OLAP structure. From the example, the following features are essential in order to satisfy the requests from users:

- Perform OLAP operations over the data
- Access to both databases without changing their structure but being able to generate the results
- Both databases have an OLAP structure in which basic OLAP operations such as AVG, SUM, COUNT can be applied as multi-level and multi-dimensionally
- Build OLAP observations in a common format
- Be able to perform data merging for building the response or to materialise it in a new database

In order to offer a solution for the above example, we need to provide a way in which the query is able to distribute to multiple databases, perform OLAP over each database and compose the results retrieved from them as presented in Figure 4.2 below.

Figure 4.2 Integrated system for collective query of semantic OLAP Databases

4.2 Conceptual framework

A key factor in successfully performing OLAP over Semantic Web data is to acknowledge the characteristics and the relationship of data. As previously introduced in Chapter 2, the two categories are: ***informational*** (dimensions are coming from node attributes) and ***topological*** (when dimensions are coming from node and edge attributes). In the case of informational data, which is also represented as a three graph, each node represents a level in a given dimension as *time*. On the other hand, topological data focuses on linking information from two different nodes (identified as dimensions) based on a connecting edge's attributes, describing a semantically richer connection of the data. Such an example is described in the case study through Household and Income dimension, where the connecting edge (property) is describing that a household *hasA* income.

Some databases may be structured using one type, (for example DB2 introduced in the previous section) while others may have a mix of structures with the information offered from different dimensions (for example DB1 from the same section).An example of a mix structure was introduced in Case Study I above and visualized in Figure 4.2.

A middleware system is needed to perform collective OLAP operations over multiple databases to store the newly generated views in a multidimensional database as well as having an expressive vocabulary to model both topological and informational structure. Even though multiple semantic OLAP databases are accessible, composing retrieved data from them is a complex process. A pipe architectural style can be designed to handle RDF and summary data that can be fed into OLAP functions to support decision making.

This work asserts that the key elements for composing such complex results are not yet fully available. Some related work, introduced in Chapter 2, has been proposed but each approach has limitations.

In order to provide OLAP functionality over multiple semantic databases, the work presented in this thesis provides a three level contribution:

- An integrated system for collective querying over multiple multidimensional databases
- An extended vocabulary for multidimensional data representation
- A materialization of semantic OLAP database capability

The conceptual framework, presented in Figure 4.3 , includes multiple layers in order to address the issues identified and discussed in the previous chapters.

On the bottom layer we have the raw data from relational databases and web data in different forms. In the case of data stored in relational databases, the layer on top of it provides multidimensional modelling of data. For the web data, there is an intermediate layer between raw data and data modelling for OLAP. This layer is described by linked data, which is a specific type of semantic web data. This layer can also be an intermediate layer between data in relational databases and the modelling layer, when

data is transformed from relational databases to linked data (W3C Working Group, 2012) before further OLAP analysis.

Figure 4.3 Conceptual Framework

Regarding the multidimensional modelling layer, the data is transformed into cubes for multidimensional models. It contains a series of different vocabularies which trigger different semantic OLAP databases, so this layer can have different representations of data for OLAP. In this work an extended representation is introduced with an enhanced vocabulary and functionalities, which are lacking in other existing vocabularies on the same support layer. The introduced vocabulary and the functionalities are briefly introduced in the next subchapters and fully described in Chapter 5 – IGOLAP Vocabulary Development, respectively Chapter 6 – Materialisation of Integrated OLAP Operators for SW Databases.

The introduced framework is a multiple layer Semantic OLAP database which is able to handle data in dimensions, levels and measures in order to respond to OLAP related queries.

The top layer in the framework provides users with interfaces to specify queries and visualise the retrieved information in relation to business intelligence or decision making. The other layers provide necessary mechanisms and functions to transform the

requests into executable syntax. The framework also increases interoperability among different semantic OLAP databases. So a query can be executed to: locate datasets; retrieve data; summarise information, and, compose semantics from various semantic OLAP databases. In order to support this functionality a pipe architecture and distributed query processing is introduced.

4.3 Architecture Overview

Figure 4.4 Components diagram, shows the components and steps needed to be included in order to provide a complete middleware system, able to offer to the user a set of OLAP functionalities over multiple Semantic Web databases.

Considering the diverse ***input data***, as presented in the Conceptual Framework, there is a need for ***data transformations***. Based on the format of the input data, the set of transformations differ. But in order to provide access over multiple databases, the ***vocabulary*** used to model and transform the input data in the required – ready to store – output data should be the same. The ***output data*** will have the same format: RDF, regardless of the chosen RDF serializations available.

In case the input data is already in RDF format, the data transformation process will include only the necessary mapping and data modelling based on the provided vocabulary.

There exist different implementations and systems available for providing data transformation into RDF data. In case of XML data there is the possibility of using Jena (The Apache Software Foundation, 2011) based implementations as well as other commercial tools. On the other hand, for relational datasets, a good solution for data transformation can be provided using mappings with Relational database (RDB) to RDF Mapping Language (R2RML) (W3C Working Group, 2012) and a selected vocabulary.

Figure 4.4 Components diagram

In regards to the storage of the RDF output data, there are also a variety of solutions available, both open source and commercial (Rohloff, Dean, Emmons, Ryder, & Sumner, 2007) (Voigt, Mitschick, & Schulz, 2012). All of these solutions are able to store graph data in RDF format and what is important, they provide a SPARQL endpoint for querying the published semantic web data. Although these solutions provide storage and querying means, they do not provide a modelling vocabulary which is a highly important component in performing OLAP operations over the data.

Figure 4.5 Components diagram with highlighted mandatory components

In conclusion, the focus in this work is to provide the necessary ***vocabulary*** to model the semantic databases' datasets and the needed ***integrated OLAP operators*** that can be integrated in different user interfaces and provide the user with OLAP functionalities over Semantic Web Databases. The additional components can be choose at the discretion, resources and needs of any party deciding to implement this middleware solution as presented in Figure 4.5. – Components Diagram with highlighted necessary components.

Figure 4.6 shows that using the same extended vocabulary to model the data can provide the possible use of integrated OLAP operators to retrieve composed data from multiple databases.

Figure 4.6 Data flow in a multiple databases scenario

As will be presented in the following subsection, the introduced operators are designed not only to retrieve and compose the data from multiple semantic web databases but also to materialize the result in an additional dataset and store it in a provided semantic database.

As the main components in this middle layer, and the ones on which this work is focusing, are the vocabulary and the integrated OLAP operators. They are briefly introduced in the following two subsections and fully described in two dedicated

chapters, Chapter 5 – IGOLAP Vocabulary Development and Chapter 6 – Materialisation of Integrated OLAP Operators for SW Databases.

4.3.1 IGOLAP Vocabulary

As mentioned in Chapter 2 – Research Background, there is existing work regarding a vocabulary for multidimensional data modelling for OLAP support. In this work it was considered that the QB vocabulary does not have sufficient capabilities to handle OLAP, but it has the adequate structure. Additionally the QB4O vocabulary is an extended version of the QB vocabulary, offering more functionality to support OLAP. Nevertheless, both vocabulary sets have missing facilities in relation to modelling two groups of data: Informational (where dimensions are coming from node attributes) and Topological (where dimensions are coming from node and edge attributes). The two groups of data are described more into details in the next Chapter – IGOLAP Vocabulary development. Their vocabulary needs to be extended and altered in order to provide full OLAP capabilities. Since an informational graph is modelled by dimensions and hierarchical levels and the topological graph is modelled in dimensions, members and defined relationships, the type of aggregations over their measures are very different. On the informational graphs the standard measure aggregations such as SUM, AVG, and COUNT are used to summarise the data, but the topological graphs require relationship type of aggregations. To design a unified semantic OLAP to handle both graphs is not trivial.

Considering that the dimensions in the topological structure do not have levels but direct members, two different classes were introduced to model it: *igolap:InfoDimension* and *igolap:TopoDimension* as subclasses of the *qb:DimensionProperty* class. Additionally the property that connects these two classes to their superclass: *igolap:dimensionType* was also introduced. The new vocabulary is presented in Figure 4.7 and explained in detail in Chapter 5.

Figure 4.7 IGOLAP vocabulary

The main required changes and additions are summarized below:

- The existing QB4OLAP had to be altered in order to handle the topological dimension. As such, the modified member concepts which has new connecting property with the introduced topological dimension was introduced.
- Alterations regarding the attribute properties have now to reflect both topological and informational dimension attributes. The informational dimension has levels which have members and the topological dimension has direct members. In order for the property *qb4o:hasAttribute* to apply to both topological and informational dimension, it has to connect the *igolap:Member* to *qb:AttributeProperty*.
- As the topological dimensions can be connected to each other through a topological property and each member of the dimension holds the property, a new property was needed in order to define those connections.

- For support in the drill-down operation, an inverse function for the parentLevel property was introduced in the presented vocabulary.

In this subchapter the Vocabulary is briefly introduced while in Chapter 5 it will be fully described and compared with the base QB4O and QB vocabularies.

4.3.2 Integrated OLAP Operators

The proposed framework is based on a set of specialized OLAP operators (Federated OLAP operators) that can operate over multiple semantic OLAP databases, merge the outputs into a common format and translate them according to the desired output which can be materialized or viewed.

The Federated OLAP operators need to interpret the requests according to a specific OLAP database in order to retrieve the data and convert it to a requested output format. The Federated OLAP operators represent an extension of the classic OLAP operations as: roll-up, dice, drill down or slice.

All the operators used in this research work are based on the traditional description formulated and agreed upon in many research approaches (Agrawal, Gupta, & Sarawagi, 1997) (Chaudhuri & Dayal, 1997) (OLAP Council, 1997).

The extension of the SPARQL querying language with federation capability is out of the scope of this research. There are other stream of works focusing on this aspect including (Buil-Aranda, Arenas, & Corcho, 2011), which focuses on the optimization techniques for querying federations. In this approach the federation targets the metadata representation aspect, and the possibility of such designed operators to access multiple semantic databases modelled using this vocabulary.

The dimension operations that are used in this approach are defined as F_Operators. These include the standard dimension operators as F_ROLL_UP, F_DICE, F_SLICE and F_DRILL. They are derived from the standard OLAP dimension operations, but they are adapted to have the necessary functions to access multiple semantic databases.

The standard OLAP measure operations are used as restriction functions in the dimension operations that include AVG for retrieving the arithmetical mean of a set of

numerical values, SUM for the sum of a set of numerical values, COUNT for the cardinality of a set of elements and MIN and MAX for the minimum and maximum element of a set of elements.

Exemplification of the F_OPERATORS through F_ROLL_UP

The F_ROLL_UP operator is briefly introduced in the next paragraphs, with a complete description being available in Chapter 6.

The F_ROLL_UP operator includes a set of processes. For the retrieval stage, the operator identifies the targeted databases, builds the SELECT operators for each database with given constraints and gathers information from multiple databases by applying the built-in operators to specific datasets. In the building stage, the CONSTRUCT operator is initiated to compose the response from the retrieved data. When the datasets retrieved by each SELECT operator are in the same format, the CONSTRUCT operator is applied directly, but if the datasets have different formats, data normalisation is performed before generating the output. In order to handle the data exchange, the F_ROLL_UP operator is described as a pipe architecture containing a CONSTRUCT operator and a number of SELECT operators. If data normalisation is required before the output is generated the third operator, the MERGE operator, is included in the F_ROLL_UP pipe construction. The MERGE operator is used to structure the partial RDF triple results from the SELECT operators using the same vocabulary for the output construction. Even though the MERGE concept has some similarity with the one in the semantic web pipes, the MERGE from semantic web pipes is a simple join of the CONSTRUCT and/or SELECT operators output without normalisation capabilities and facilities to support OLAP. In the F_Operators case, the MERGE operator is merging the different databases by merging the namespaces prefixes, as presented in Chapter 6, and also in Figure 6.4 Sequence diagram of F_Operators.

Since F_OPERATOR's are designed to access one or more than one OLAP database, they require a set of arguments in order to interpret the requests. Based on the arguments received, F_ROLL_UP distinguish between:

- single or multiple database access;

- formatted or unformatted output;
- request for view or request for materialization of the output, and so on.

This means that the parameters can be divided into two main categories: the mandatory and the optional ones (e.g. materialised or immaterialised output represents an optional parameter). The mandatory parameters that need to be passed on are: location of accessed Semantic Web OLAP implementation(s) (URIs or IRIs), dimensions (and dimension level for F_ROLL_UP) and some others.

4.4 Summary

This chapter introduced the two case studies that provided the context and the data for this research. The use cases contain time series data enhanced by semantics which is a challenge for the traditional RDMS systems to model without missing out existing semantic or semantical information that could be added later. While in the energy consumption use case was illustrated that there is already available data that is rich in semantics which can be lost in the case of modelling for relational databases, the second use case shows how data could be enriched at any point of time with additional information coming from additional databases. But in order to be able to aggregate and to retrieve contextual insights from these additional resources, this information should be modelled using also a multidimensional modelling vocabulary.

Further this chapter contains the conceptual framework as well as the architectural overview of the work described in this thesis, providing the context to understand the importance of the contribution of this research work. The main to components that this thesis focuses on are briefly introduced in this chapter, giving the possibility to the reader to investigate in detail the implementation and construction of these components in the two chapters to follow.

Chapter 5 – IGOLAP Vocabulary Development

5.1 Introduction

With the increasing volume of data available, integration of data from disparate data sources is becoming harder to achieve, especially in the case of data from the semantic web. Increasingly available repositories with semantic data provide the content for further analyses for better decision making support. But, in order to provide the required reports and data analysis, a certain type of data warehousing over semantic web data needs to be provided.

In the previous chapters the existence of a set of vocabularies for publishing RDF statistical data was presented. From this set, two vocabularies provide a good basis for modelling semantic web data for OLAP: Data Cube Vocabulary (Tennison & TSO, 2011)(denoted QB) and QB4OLAP (Etcheverry & Vaisman, 2012), with the latter being an extension of the first. Nevertheless, while QB follows statistical data models and QB4OLAP follows the classic multidimensional models for OLAP, neither consider the variety of characteristics of the two categories of data by which semantic web data can be classified.

The following subsections introduce why there is a need to extend the available vocabularies and describe the difference between the two classifications of semantic web data used in this research work, namely, informational and topological data. As it was mentioned in the previous chapter, the current chapter also explains the development and the additions in the IGOLAP vocabulary in a detailed way.

5.2 Identification of a base vocabulary

The first concept used in performing OLAP is the capability to model data multi-dimensionally – into data cubes. Additionally, in the semantic web context, it is particularly important to be able to publish these data, which is commonly done through RDF. In this context and considering the research conducted and depicted in Chapter 2 – Research Background, the best candidate was considered to be QB4OLAP, which is itself an extension of the RDF data cube vocabulary (QB). On one hand the QB vocabulary does not have sufficient capabilities to handle OLAP, but it has adequate structure. On the other hand, although QB4OLAP vocabulary is an extended version of

QB, offering more functionality to support OLAP it only addresses the informational type of data.

Although there is other existing work that was considered and presented in Chapter 2, those streams of work do not provide a sufficiently powerful vocabulary. As shown in that research background chapter, the research work identified to model multidimensional semantic data is addressing only informational or topological semantic data, but not both; and most available online semantic web data contains both.

The concepts of informational and topological data are explained in the following section of this chapter, while identifying the additional limitations of the base vocabulary identified in this section.

5.2.1 RDF data cube vocabulary and QB4OLAP overview

The RDF data cube vocabulary (QB) covers the initial set of requirements, presented previously, for statistical data and metadata which is published on the web. An overview of the QB vocabulary is presented in Figure 5.1., where the RDF classes and properties with which the vocabulary operates can be identified. The representation uses capitalized terms for classes and non-capitalized terms for properties.

Further, the vocabulary is targeting statistical data by use of concepts from the model which is the ISO standard for data exchange among organisations. Additionally QB imports concepts and properties from the Simple Knowledge Organization System (SKOS) (ArticulateSoftware, n.d.) vocabulary. These classes, alongside the other external vocabularies' classes and properties, are depicted in Figure 5.1. in light grey.

As previously introduced, the QB vocabulary is designed to address statistical data with its defined support data structure definition (DSD), defined as an instance of the class *qb:DataStructureDefintion*. This is used to specify the schema of a dataset, which is an instance of the class *qb:DataSet*. Different datasets can share the same structure definition. In order to represent dimensions, measures and attributes, a DSD contains component properties to represent these, and they are explicitly named, as seen in Figure 5.1. As such these are defined using the component property and blank nodes.

Additionally the vocabulary provides means to represent observations (qb:Observation), which can be then grouped in datasets through the *qb:dataset* property. Instances of the *qb:DimensionProperty* are properties which can link an observation to a value in a DSD dimension. The *qb:MeasureProperty* and the *qb:AttributeProperty* work in a similar way in regard to observations and a set of measures or attributes.

Figure 5.1 QB vocabulary (Tennison & TSO, 2011)

With all these, QB allows the representation of hierarchical relationships between members of dimension levels, making use of the SKOS vocabulary, but it cannot represent a multidimensional schema. More insights with regard to multidimensional representation issues are provided in the following section alongside other encountered issues for both QB and QB4OLAP.

The QB4OLAP vocabulary aims to counteract the shortcomings of the QB vocabulary in regard to multidimensional data representation and the possibilities for performing OLAP standard operations over the modelled data. In order to be able to overcome these limitations QB4OLAP introduces a set of new classes and properties to model the relationships and support the mentioned functionalities.

Figure 5.2 QB4OLAP vocabulary

The new additions to the initial QB vocabulary extended it with dimension levels (introducing hierarchies on the dimension) and level members as well as providing aggregating functions which are linked to measures. The additions presented are highlighted in dark gray, and as it is observed from the Figure 5.2 they do not affect the structure of the QB vocabulary. Thus the existing applications developed with it are not invalidated.

In Table 5.1 the set of classes and properties available in the QB vocabulary and the additions of QB4OLAP are depicted. As the QB4OLAP is an extension of the former vocabulary, it contains also all the classes and properties of the QB Vocabulary but this is not repeated in the table, only the new concepts introduced.

Table 5.1 Classes and properties comparison between QB and QB4O

5.3 Identification of the limitations of the base vocabulary

In the previous section the vocabularies chosen in this research work as a base vocabulary were introduced. As mentioned in the research background (Chapter 2) and previously in this chapter, these have a set of limitations that can be considered in three main categories:

- handling of both structures of data from the semantic web: topological and informational
- the capability to represent the semantic web data multi-dimensionally
- supporting the construction of specialized OLAP operators

5.3.1 Informational and topological dimensions

As briefly introduced in the previous chapters and sections, there is a stream of work that addressed the fact that graph network data, such as semantic web data, comes across in informational and topological types of graphs. In an approach to multi-dimensionally represent these graphs, it is necessary to consider the two different types of dimensions that these graph types generate:

- ***Informational dimension***, where the dimensions are generated from the graph node attributes. In an informational dimension it will be possible to have different hierarchical levels of the dimension. The representation of this type of data would resemble a tree-type of graph, in which going from the top node to the last leaf in the graph, each level would represent a more detailed and context restricted view of the data.

- These tree-graph data usually are available in online statistical type of data; one such example would be government demographic data delivered by the census bureau. In this case it would be easy to identify *time* dimension with hierarchal levels as: year, quarter, and month or *location* dimension with levels as country, county, and city. Additionally, measures are a given year, quarter or month of a year. In this example, taking the time dimension, each node would be a level and the edge connecting *year node (or level)* to *quarter node*, and this could be *parentLevel*.
- ***Topological dimension*** represents dimensions that come from a combination of the node and edge attributes. This combination of node and edge attribute delivers a perspective that is relevant for the topological aspect of the data graph. Taking the example run through this thesis, in which would had *income* as a dimension of the graph. Income can only have measures defined by different concrete income values or income ranges. But it would have a connecting edge or property to another dimension, for example *household*. Since each dimension can have directly connecting property with some other such dimensions, the resulting visualisation of a graph containing topological dimensions would be a network graph.

Real world online data is available in a mix of topological and informational graphs. For data available in this structure, mostly semantic data, it is challenging to apply generic OLAP operators. An example of such data is visualised in Figure 5.3, derived from Case Study II, introduced in the previous chapter.

Figure 5.3 Identified topological and informational dimensions in collected data

As previously introduced, an informational graph is modelled by dimensions and hierarchical levels and the topological graph is modelled in dimensions, members and defined relationships, the types of aggregation over their measures are very different. On the informational graphs the standard measure aggregations such as SUM, AVG, and COUNT are used to summarise the data, but the topological graphs require relationship types of aggregation. To design a unified semantic OLAP to handle both graphs is not trivial.

Continuing the example provided in the previous Chapter in the Case Study II, as seen in Figure 5.3as well, it is now understood as if in a database information about energy consumption is available and also information regarding the household and appliances that produce this consumption. It would be necessary for this information to be modelled using both topological and informational dimensions. These two types of dimensions have relationships that connect them. These relationships are described through new properties designed to connect topological and informational dimensions.

In Figure 5.3, it is observed as Household, Income, Appliances, are topological dimensions of the collected data. These dimensions do not have hierarchical levels, but the relationships connecting different such nodes have attributes that determine the dimension that they represent. As such the attributes of both the nodes and their relationships are encapsulated in the attributes of the dimension. On the other hand, information regarding actual consumption, including dimensions such as Location and Time and their hierarchical levels, and the measure of Consumption, represent typical cases of informational dimensions with classical multidimensional attributes. The latter types of dimensions are the ones that can be modelled using QB4OLAP as well. Another aspect is that additional properties connecting the topological dimensions to the informational ones are not available either in the chosen base vocabularies or in the available literature presented in Chapter 2.

5.3.2 Multidimensional data representation

With regard to the multidimensional data representation capability, as also identified in (Vaisman & Zimányi, 2014), the QB vocabulary allows for the representation of hierarchical relationships between members of dimension levels. This is mostly done

through using the SKOS vocabulary. With all this, by using the property *skos:hasTopConcept* to define the highest hierarchy level in the dimension and making use of the *skos:narrower* property to allow navigation from higher to lower levels in the dimension's hierarchy, the QB vocabulary has an opposite direction data retrieval model as compared with the majority of the common OLAP operations. In the case of the OLAP operations, with the exception of drill-down, the dimension is traversed from the lower to the higher granularity of the dimension's levels.

Addressing this aspect, the QB4OLAP model has the property *qb4o:parentLevel* to define the relationship between instances of the class *qb4o:LevelProperty*. As well, in contrast with QB, it uses the *skos:broader* property to define the parent of a level member. Also the property *qb4o:inLevel* indicates to which level a level member belongs. With the additions that QB4OLAP provides, however, it still is sufficiently compatible to handle observations defined using the QB vocabulary and add these to data expressed using the QB4OLAP vocabulary.

But while on one hand QB had a top down approach to navigating to the levels of a dimension, using the SKOS vocabulary, on the other hand QB4OLAP introduced only a property to handle a bottom up approach – *qb4o:parentLevel*. While this property addresses the most common OLAP operators' data retrieval direction, it doesn't directly support the drill-down OLAP operator. For this, an inverse property needs to be defined to the *qb4o:parentLevel*. This gives the possibility to use any direction of navigation as per considered appropriate for the desired operator. Considering the OWL syntax defined in (W3C, 2004) and the previous definition oft he *qb4o:parentLevel* propriety, this is defined as:

```
<owl:ObjectProperty rdf:ID="childLevel">
    <owl:inverseOf rdf:resource="&qb4o;parentlLevel" />
</owl:ObjectProperty>
```

5.3.3 OLAP operations of QB and QB4OLAP

In the previous section there was brief mention of the possible operators for OLAP. This is mostly due to the fact that OLAP operators rely on the multidimensional

representation of the data. Common OLAP operators have great limitations in regard to data represented using the QB vocabulary, with the exception of the Dicing operator they cannot be directly implemented. This latter can be implemented with the use of the FILTER clause in SPARQL. With regard to the other operators, due to the limited support for the dimension hierarchy, they are not supported and the reasons for this can be summarized as follows. In the case of the roll-up operations this is due to the direction in which the dimension is traversed and the missing modelling of the levels and relationships between level members. Additionally for both drilling-down and slicing, the missing modelling of the aggregation functions for a given measure means that the implementation of the mentioned operators is not supported. This is due to the fact that for OLAP tools generally there is an association of each measure to an aggregation function to support the standard roll-up, drill-down and slicing operators.

Although the QB4OLAP makes it possible to implement the slice, roll-up and dice operators, the drill-down operator is not supported. Furthermore, these operators can address only data modelled by informational dimensions and as such cannot be expressive enough to address the entire spectrum of information from semantic web data. Furthermore, not only can topological and informational data not be simultaneously handled by these operators, but the possibility of using the mentioned OLAP operators over multiple databases, to create further aggregations, is not supported.

5.4 IGOLAP Vocabulary and possible OLAP Operations

In the previous section of the current chapter, the base vocabulary used and the limitations that this vocabulary has, was identified. In this section there is an introduction to the additions needed in regards to the main three limitations identified:

- mix informational and topological data handling,
- multidimensional representation,
- support for OLAP operations.

As noted, the existing vocabularies modelling multidimensional data for OLAP activity can only be considered as base vocabularies. It was previously identified that

both QB and QB4OLAP vocabulary sets have missing facilities in relation to modelling two groups of data: informational and topological. These aspects are considered in the proposed vocabulary's additions in order to provide full OLAP capabilities.

Nonetheless, as the purpose of this work is to address OLAP functionalities, in section 5.4.3 the OLAP operations that are now available due to the new set of vocabulary's additions are identified.

5.4.1 Additions to the base and development of IGOLAP Vocabulary

Considering that the dimensions in the topological structure do not have levels but direct members, two different classes are introduced to model it: ***igolap:InfoDimension*** and ***igolap:TopoDimension*** are subclasses of the qb:DimensionProperty class. The property that connects these two classes to their superclass is***: igolap:dimensionType***. The new vocabulary is presented in Figure 5.4 and the comparison of the vocabularies' capabilities is presented in Table 5.2. The set of classes and properties introduced in the IGOLAP Vocabulary are depicted in a light blue colour in the mentioned figure.

The existing *qb4o:LevelMember* had to be altered in order to handle the topological dimension. The modified ***igolap:Member*** is introduced; which, while keeping the connection with *qb4o:LevelProperty* also has a new connecting property to the topological dimension: ***igolap:ofDimension***. Since both topological and informational dimension have attributes, the property *qb4o:hasAttribute* had to be altered to reflect this.

The informational dimension has levels which have members and the topological dimension has direct members. In order for the property *qb4o:hasAttribute* to apply to both topological and informational dimension, it has to connect the *igolap:Member* to *qb:AttributeProperty*.

Figure 5.4 IGOLAP vocabulary

The topological dimensions can be connected to each other through a topological property and each member of the dimension holds the property. To define those connections the *igolap:topoDConnectedTo* property has been introduced.

QB4OLAP introduces the *qb4o:parentLevel* property which connects levels and can support the roll-up operation, but in order to offer a better support for the Drill-down operation this proposal introduces the ***igolap:childLevel*** property which also connects levels and is an inverse function to *qb4o:parentLevel*, but not symmetric. The meaning of this is that, while the *qb4o:parentLevel* property delivers a parent to child relationship, which is a ***1 to many*** type of relationship, from child to parent is a ***many to 1*** type of relationship which is explicitly provided through the *igolap:childLevel* property.

Table 5.2 Classes and properties of IGOLAP in addition to QB and QB4OLAP vocabularies

OWL materialization of the vocabulary is presented in full below. As the IGOLAP is based on the QB4OLAP Vocabulary this extract included both in order to maintain the context. Highlighted in light blue are the additions and changes to the original QB4OLAP. As no changes were added to the QB vocabulary, this is provided in full alongside the original QB4OLAP vocabulary in the Appendix A.

```
@prefix igolap: <http://topublish.org/igolap#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .

http://topublish.org/igolap a owl:Ontology;
owl:versionInfo "0.1";
rdfs:label "The IGOLAP vocabulary";
rdfs:comment "This vocabulary allows to publish and operate with OLAP cubes in RDF
   containing both topological and informational data";
dcterms:created "2013-01-17"^^xsd:date;
dcterms:modified "2013-08-15"^^xsd:date;
dcterms:title "Vocabulary for publishing topologicaland informational OLAP data cubes";
dcterms:license <http://www.opendatacommons.org/licenses/pddl/1.0/> ;
dcterms:contributor [foaf:mbox "mateia@uni.coventry.ac.uk"];
.

# --- Levels and Level members --------------------------

qb4o:LevelProperty a rdfs:Class, owl:Class;
rdfs:label "Level property"@en;
rdfs:comment "The class of components which represent the levels of a dimension"@en;
rdfs:subClassOf qb:ComponentProperty;
rdfs:subClassOf qb:CodedProperty;
rdfs:isDefinedBy <http://purl.org/olap>;
.

igolap:InfoDimension a rdfs:Class, owl:Class;
rdfs:label "Informational Dimension"@en;
rdfs:comment "The class of Dimension Property for Informational Graphs"@en;
rdfs:subClassOf qb:DimensionProperty;
.

igolap:TopoDimension a rdfs:Class, owl:Class;
rdfs:label "Topological Dimension"@en;
rdfs:comment "The class of Dimension Property for Topological Graphs"@en;
rdfs:subClassOf qb:DimensionProperty;
.

igolap:Member a rdfs:Class, owl:Class;
rdfs:label "Level member"@en;
rdfs:comment "The class of components which represent the members of a level in a
   InfoDImension or the members of TopoDimensions"@en;
rdfs:subClassOf skos:Concept;
.

qb4o:level a rdf:Property, owl:ObjectProperty;
rdfs:label "level"@en;
rdfs:comment "An alternative to qb:componentProperty which makes explicit that the
   component is a level"@en;
rdfs:subPropertyOf qb:componentProperty;
rdfs:range qb4o:LevelProperty;
rdfs:isDefinedBy <http://purl.org/olap>;
.

igolap:member a rdf:Property, owl:ObjectProperty;
rdfs:label "member"@en;
rdfs:comment "An alternative to qb:componentProperty which makes explicit that the
   component is a member, which is relevant for members in Topodimensions"@en;
rdfs:subPropertyOf qb:componentProperty;
rdfs:range igolap:Member;
.

igolap:dimensionType a rdf:Property,owl:ObjectProperty;
rdfs:label "type of dimension"@en;
rdfs:comment "Indicates the type of dimension that the Dimension Property is"@en;
rdfs:range qb:DimensionProperty;
rdfs:domain igolap:TopoDimension;
rdfs:domain igolap:InfoDimension;
.

igolap:topoDConnectedTo a rdf:Property, owl:ObjectProperty;
rdfs:label "topological dimensions connections"@en;
```

```
rdfs:comment "Indicates to which other topological dimension a topological dimension is
   connected"@en;
rdfs:range igolap:TopoDimension;
rdfs:domain igolap:TopoDimension;
.

qb4o:inDimension a rdf:Property, owl:ObjectProperty;
rdfs:label "level in dimension"@en;
rdfs:comment "Indicates to which dimension the level belongs, applicable only to
   InfoDimensions"@en;
rdfs:range qb4o:LevelProperty;
//this is not applicable in igolap anymore as only InfoDimensions have levels
   rdfs:domain qb:DimensionProperty;
rdfs:domain igolap:InfoDimension;
.

igolap:ofDimension a rdf:Property,owl:ObjectProperty;
rdfs:label "member in dimensions"@en;
rdfs:comment "Indicates for topological dimensions, that a member belongs to a dimension
   without levels "@en;
rdfs:range igolap:Member;
rdfs:domain igolap:TopoDimension;
.

qb4o:inLevel a rdf:Property, owl:ObjectProperty;
rdfs:label "level member in level"@en;
rdfs:comment "Indicates to which level the level member belongs"@en;
rdfs:range igolap:Member;
rdfs:domain qb4o:LevelProperty;
rdfs:isDefinedBy <http://purl.org/olap>;
.

qb4o:parentLevel a rdf:Property, owl:ObjectProperty;
rdfs:label "is parent of"@en;
rdfs:comment "Indicates which is the parent level of each level"@en;
rdfs:range qb4o:LevelProperty;
rdfs:domain qb4o:LevelProperty;
rdfs:isDefinedBy <http://purl.org/olap>;
.

igolap:childLevel a rdf:Property, owl:ObjectProperty;
rdfs:label "is child of"@en;
rdfs:comment "Indicates which is the child level of a level"@en;
rdfs:range qb4o:LevelProperty;
rdfs:domain qb4o:LevelProperty;
owl:inverseOf qb4o:parentLevel;
.
# --- Aggregate Functions ----------------------------

qb4o:AggregateFunction a rdfs:Class, owl:Class;
rdfs:label "Aggregate function"@en;
rdfs:comment "The class of components which represent aggregate functions that are
   applied to compute measure aggregate values"@en;
rdfs:isDefinedBy <http://purl.org/olap>;
.

qb4o:sum a qb4o:AggregateFunction;
rdfs:label "SUM"@en;
rdfs:comment "Returns the numeric value obtained by adding a set of numeric values."@en;
owl:sameAs dbpedia:Summation;
.

qb4o:avg a qb4o:AggregateFunction;
rdfs:label "AVG"@en;
rdfs:comment "Returns the arithmetic mean of a set of numeric values."@en;
owl:sameAs dbpedia:Average;
.

qb4o:count a qb4o:AggregateFunction;
rdfs:label "COUNT"@en;
rdfs:comment "Returns the number of elements in a set of elements (the cardinality of
   the set)."@en;
owl:sameAs dbpedia:Counting;
```

```
.

qb4o:min a qb4o:AggregateFunction;
rdfs:label "MIN"@en;
rdfs:comment "Returns the minimum element in a set of elements, where a partial order is
   defined."@en;
owl:sameAs dbpedia:Min;
.

qb4o:max a qb4o:AggregateFunction;
rdfs:label "MAX"@en;
rdfs:comment "Returns the maximum element in a set of elements, where a partial order is
   defined."@en;
owl:sameAs dbpedia:Max;
.

qb4o:hasAggregateFunction a rdf:Property, owl:ObjectProperty;
rdfs:label "has aggregate function"@en;
rdfs:comment "Indicates which aggregate function has to be applied to obtain measure
   aggregate values, for a certain measure in a cube"@en;
rdfs:range qb:ComponentSpecification;
rdfs:domain qb4o:AggregateFunction;
rdfs:isDefinedBy <http://purl.org/olap>;
.
```

As presented in this section, the newly introduced classes and properties directly address the QB and QB4OLAP vocabularies' limitations described in the previous section. Furthermore, these additions also address the issues from the current literature in regard to OLAP capabilities over multidimensional semantic web data; as identified in Chapter 2 – Research Background

.

5.4.2 Usage of the IGOLAP Vocabulary

Using Case Study II from the Chapter 4, it is shown how both informational and topological structures can be implemented using the IGOLAP vocabulary elements. The content and structure of DB1 in the scenario are described in Figure 4.1 and Figure 4.2 and it contains curated data of both types of structure. In this example the datasets' prefixes are omitted.

The code extract – Extract-1 below – shows the structure of the informational dimensions of an energy consumption database, DB1. It shows the representation of the time and location dimension structures as well as instances of the location. The structure of the informational dimension is very similar to the QB4OLAP vocabulary, but the *igolap:childLevel* in association with *qb4o:parentLevel* property give the potential for

bidirectional navigation in order to support both roll-up and drill-down OLAP operations.

Extract 1: Informational Dimensions – Time and Location Schema – and Location Instances:

```
e:location a igolap:InfoDimension.

e:country a qb4o:LevelProperty;
   qb4o:inDimension e:location;
   igolap:childLevel e:firstAdminDivision.
e:firstAdminDivision a qb4o:LevelProperty;
   qb4o:inDimension e:location;
   igolap:childLevel e:secondAdminDivision;
   qb4o:parentLevel e:country.
e:secondAdminDivision a qb4o:LevelProperty;
   qb4o:inDimension e:location;
   igolap:childLevel e:city;
   qb4o:parentLevel e:firstAdminDivision.
e:city a qb4o:LevelProperty;
   qb4o:inDimension e:location;
   qb4o:parentLevel e:secondAdminDivision.

e:time a igolap:InfoDimension

e:year a qb4o:LevelProperty;
   qb4o:inDimension e:time;
   igolap:childLevel e:quarter.
e:quarter a qb4o:LevelProperty;
   qb4o:inDimension e:time;
   igolap:childLevel e:month;
   qb4o:parentLevel e:year.
e:month a qb4o:LevelProperty;
   qb4o:inDimension e:time;
   igolap:childLevel e:day;
   qb4o:parentLevel e:quarter.
e:day a qb4o:LevelProperty;
   qb4o:inDimension e:time;
   qb4o:parentLevel e:month.

gn:2635167 a igolap:Member;
   qb4o:inLevel e:country;
   rdfs:label "United Kingdom@en";
   igolap:childLevel gn:6269131.
gn:6269131 a igolap:Member;
   qb4o:inLevel e:firstAdminDivision;
   rdfs:label "England@en";
   igolap:childLevel gn:3333134;
   igolap:childLevel gn:3333125.
gn:3333134 a igolap:Member;
   qb4o:inLevel e:secondAdminDivision;
   rdfs:label "City of Bristol@en";
   igolap:childLevel gn:2654675.
gn:2654675 a igolap:Member;
   qb4o:inLevel e:city;
   rdfs:label "Bristol@en";
   qb4o:parentLevel gn:3333134;
gn:3333125 a igolap:Member;
   qb4o:inLevel e:secondAdminDivision;
   rdfs:label "City and Boroughof Birmingham@en";
   igolap:childLevel gn:2655603.
gn:2655603 a igolap:Member;
   qb4o:inLevel e:city;
   rdfs:label "Bristol@en";
   qb4o:parentLevel gn:3333125;
```

Extract 2 below, shows the representation of the topological dimensions that include the three dimensions: income, household, and, appliances. These dimensions are connected by a connecting property. These dimensions do not have a well-defined hierarchical structure but they define aggregations based on common attributes (e.g. number of bedrooms, or number of inhabitants, in the household dimension). Shown in Extract 2 are the instances of each dimension, with the necessary attributes to populate DB1; producing possible observations.

Extract 2: Topological Dimensions: Income, Household, Appliance and instances

```
te:incomeRange a igolap:TopoDimension;
   igolap:TopoDConnectedTo te:household.

te:ukI04 a igolap:Member;
   igolap:ofDimension te:incomeRange;
   igolap:TopoDConnectedTo te:hhBrs01;
   igolap:TopoDConnectedTo te:hhBhm73;
   te:min 25000;
   te:max 35000;
   te:currency "GBP".

te:household a igolap:TopoDimension;
   igolap:TopoDConnectedTo te:incomeRange;
   igolap:TopoDConnectedTo te:appliances.

te:hhBrs01 a igolap:Member;
   igolap:ofDimension te:household;
   igolap:TopoDConnectedTo te:ukI04;
   igolap:TopoDConnectedTo te:fridge_ZZRB934FW2Brs01;
   igolap:TopoDConnectedTo te:tv_SUE22D5003Brs01;
   te:hasCity gn:2654675;
   te:hasPostCode "BS35";
   te:size     57;
   te:bedrooms  3;
   te:bathrooms 2;
   te:houseType "detached";
   te:built 1987.

te:appliance a igolap:TopoDimension;
   igolap:TopoDConnectedTo te:household.

te:fridge_ZZRB934FW2Brs01 a igolap:Member;
   igolap:ofDimension te:appliances;
   igolap:TopoDConnectedTo te:hhBrs01;
   te:hasModel "ZZRB934FW2";
   te:hasPostCode "Zanussi";
   te:consumption e:fridge_ZZRB934FW2_20090522;
   te:consumption e:fridge_ZZRB934FW2_20090523;
   te:consumption e:fridge_ZZRB934FW2_20090527;
   te:consumption e:fridge_ZZRB934FW2_20090612;
   te:consumption e:fridge_ZZRB934FW2_20090719.

te:tv_SUE22D500Brs01 a igolap:Member;
   igolap:ofDimension te:appliances;
   igolap:TopoDConnectedTo te:hhBrs01;
   te:hasModel "SUE22D500";
   te:hasPostCode "Samsung";
   te:consumption e:tv_SUE22D500_20090522;
   te:consumption e:tv_SUE22D500_20090523;
   te:consumption e:tv_SUE22D500_20090527;
   te:consumption e:tv_SUE22D500_20090612;
   te:consumption e:tv_SUE22D500_20090719.
```

Subsequently there is the definition of the measure for the database and the included attribute properties which are used in generating observations. In Extract-3, there is a definition of a measure for consumption and its attribute which is the measurement unit (e.g. kWh).

Extract 3: Observations structure definitions and instances

```
e:consumptionYCity a qb:DataStructureDefintion;
   qb:component [qb4o:level e:year];
   qb:component [qb4o:level e:city];
   qb:component [qb4o:measure e:consumption;
            qb4o:hasAggregateFunction qb4o:sum];
   qb:component [qb:attribute e:electricMeasureUnit].

e:datasetYCC a qb:DataSet;
   rdfs:label "Yearly consumption in a city@en";
   qb:structure e:consumptionYCity.

e:consumptionMHAC a qb:DataStructureDefinition;
   qb:component [qb4o:level e:month];
   qb:component [igolap:member te:household];
   qb:component [igolap:member e:appliance];
   qb:component [qb4o:member e:consumption;
            qb4o:hasAggregateFunction qb4o:sum];
   qb:component [qb:attribute e:electricMeasureUnit].

e:datasetMHAC a qb:DataSet;
   rdfs:label "Monthly consumption of household by appliance@";
   qb:structure e:consumptioMHAC.

e:consumption a qb:MeasureProperty.

e:electricMeasureUnit a qb:AttributeProperty.

e:kWh a e:electricMeasureUnit;
   rdfs:label "Kilowatt-hour@en".

e:o1 a qb:Observation;
   qb:dataSet e:datasetYCC;
   e:year db:2009;
   e:city gn:2654675;
   e:consumption 60000000;
   e:electricMeasureUnit e:kWh.

e:o2 a qb:Observation;
   qb:dataSet e:datasetMHC;
   e:month tl:05_2009;
   te:household te:hhBrs01;
   te:appliance te:fridge ZZRB934FW2Brs01;
   e:consumption 70;
   e:electricMeasureUnit e:kWh.
```

The structures, instances, measures and attributes mentioned are used to obtain different observations over both topological and informational dimensions, as shown in Extract 3 above. Additional constraints over topological dimensions' attributes and/or different levels of the informational dimensions are used to structure the observations.

Assume that the proposed system needs to satisfy a query in regards to the yearly energy consumption for a specific household and location. In this case it is essential to

retrieve the household information from different topological dimensions and measures like: income, property price range, household appliances or number of bedrooms. This information in conjunction with informational dimensions like time and location on different aggregation levels – as city and monthly – can deliver a complex answer on different levels of aggregation.

Furthermore, aggregations over multiple databases with topological and informational dimension, deliver the expected completeness of the answer. In this example, DB1 can only provide partial information to answer this query, such as consumption of different appliances in a household. DB2, containing informational dimensions regarding properties' market values in different years, based on the location and property layout details, can offer the complimentary information. This requires federated OLAP operations to retrieve, summarise and compose data from multiple semantic databases.

But the dimensions from different databases can have mismatched structure such as different level of detail in modelling. . The *location* dimension in DB2 has a level called *area* which is a smaller division of *city* in DB1. Another mismatch in structure among them is the existence of a redundant level *secondAdministrationDivision* from DB1. Extract 4 provides the structure and an instance of an *area* level observation.

Extract 4: Structure and instance of DB2's location dimension level

```
z:area a qb4o:LevelProperty;
   qb4o:inDimension z:location;
   qb4o:parentLevel z:city.
z:BS3 a qb4o:member;
   qb4o:inLevel z:area;
   rdfs:label "Area covering Bristol BS3 postcode@";
   qb4o:parentLevel gn:2654675.
```

Other dimensions used in this database include *time* and the property (*home*) with the property's *price* as measure. Extract 5 shows observations over the price of a certain type of property, based on the *year* and the layout of the property.

Extract 5: Observation instance of DB2:

```
z:priceByAreaProperty a qb:DataStructureDefinition;
   qb:component [qb4o:level z:area];
   qb:component [qb4o:level z:Year];
   qb:component [qb4o:level z:bedno];
```

```
   qb:component [qb4o:measure z:propertyPrice;
            qb4o:hasAggregateFunction qb4o:avg];
   qb:component [qb:attribute z:currency];

z:datasetPriceAreaProperty a db:DataSet;
   rdfs:label "Properties price by Area, nuumber of bedrooms and year@en";
   qb:structure z:priceByAreaProperty.

z:obbsBS3B1Y2009 a qb:Observation;
   qb:dataSet z:datasetPriceAreaProperty;
   z:area z:BS3;
   z:year db:2009;
   z:bedno 1;
   z:propertyPrice 125000;
   z;currency z:GBP.
```

Extract 6 shows observation structure over two databases with a common dimension: *time*. In this observation consumption information, household information and the property's market value are combined over the common dimension.

Extract 6: Observation structure over multiple databases (DB1 and DB2)

```
d:consByYearHousehold a qb:DataStructureDefinition;
   qb:component [qb4o:level z:city];
   qb:component [qb4o:level e:Year];
   qb:component [igolap:topoDImension te;household];
   qb:component [qb4o:measure e:consumption;
            qb4o:hasAggregateFunction qb4o:sum];
   qb:component [qb:attribute e:electricMeasureUnit];

d:datasetConsByYearHouseholdCMBI a db:DataSet;
   rdfs:label "Yearly consumption for households from a specific area by number of
   bedrooms and inhabitants and restricted by market value@en"
   qb:structure d:consByYearHousehold
```

These standalone observations from both databases give useful overviews of the data but their combination can provide the complete output to the query. As an example: yearly energy consumption of households is obtained by performing roll up operations over these databases from and to different levels.

5.4.3 OLAP Operations over IGOLAP Vocabulary

The proposed approach targets the capability that will provide a collective querying operation over semantic web databases. As such, the development also offers a set of specialized OLAP operators (Federated OLAP operators) that can operate over multiple semantic OLAP databases, merge the outputs into a common format and translate them according to the desired output; which can be materialized or viewed. The previously introduced vocabulary supports the possibility of applying these operators over the modelled data.

The federated OLAP operators need to interpret the requests according to a specific OLAP database in order to retrieve the data and convert it to a requested output format. These operators represent an extension of the classic OLAP operations: roll-up, dice, drill-down or slice. As such, traditional definition presented in OLAP Fundamentals are used to define the operatos over both topological and informational graphs.

5.4.3.1 Roll-up

The Roll-up operation was defined in this work with the following definition:

***Definition**: A roll-up operation assumes a data summarization inside a given cube alongside a given dimension such as in a given Cube C, a dimension $D \in C$ and a dimension level $lu \in D$, the Roll-up(C,D, lu) will return a new cube C' where measures are aggregated along D up to the level lu.*

Taking this definition for the required OLAP capability in an Informational Network and using

Figure 2.3 describing the traditional Roll-up operation, and Figure 5.5 proving the adaptation of Roll-up for topological data, provide a complementary solution on how a Roll-up operator can be applied to semantic web data.

The Roll-up operator is designed as an aggregation operator, operating over a cube by climbing up a concept hierarchy for a dimension and performing a dimension reduction.

In order to climb the dimension's hierarchy, the roll-up operator needs to use the initial hierarchy concept in order to be able to aggregate by ascending the dimension's hierarchy from the current level of the data to the desired level of aggregation. By performing a roll-up operation, one or more dimensions are not presented in the newly generated observation.

In the case of topological dimensions, these do not have hierarchy levels and as such the aggregation happens based on the dimensions' connection relationship to another topological dimension.

Figure 5.5 Topological roll-up in information networks (reproduced from (Qu, et al., 2011))

5.4.3.2 Drill down

Drill down is considered to be the reverse of Roll-up and assumes the disaggregation on a previously stored aggregation.

Furthermore the drill-down is performed by navigating through the hierarchy of a dimension from a less detailed level towards a more detailed data representation level. A generic definition of the drill down operator is provided below:

Definition*: A drill-down operation assumes a data dissaggregation inside a given cube alongside a given dimension such as in a given Cube C, a dimension D ∈ C and a dimension level lu∈ D, the Drill-down(C,D, lu) will return a new cube C', where measures are the values used to aggregate the measures on level lt, where lt>lu∈ D, along D down to the level lu.*

Although the drill-down operators is a reverse roll-up operator, while roll-up applies aggregated functions to generate the new observations, for the accuracy of the drill-down operators, the initial observations dataset from which the rolled-up ones were made needs also to be available for retrieval. In conclusion, while Roll_up and F_roll_up rely on aggregated functions, the drill-down relies on paths and hierarchical levels connections.

Since in the case of topological dimensions there is no stable mapping of one-to-one or one-to-many but it also can contain the many-to-many type of cardinality between dimensions.

In this case, navigating from one topological dimension to others, will not necessarily deliver a reduced or detailed view. In this case, the drill-down and roll-up OLAP operations have the same output in terms of the level of details.

There is one main difference though, it is that the measures for the topological dimension that we are navigating to (drilling towards), need to exist already, since the drill operator doesn't support aggregate functions on the available measures. This is due to the general description of the operator, where the focus is to find out from which measures the current one was built.

5.4.3.3 Slice

A slice operation in an OLAP understanding delivers a selection of a particular dimension from a given cube in order to provide a sub-cube.

***Definition**: Slice operation receives a cube C, a dimension $D \in C$ and a member M in level $l \in D$ and returns a sub cube C', with the same schema except that all other members in level $l \in D$ are removed.*

Since this operation addresses the members of a given level or dimension and not the levels themselves, and the TopoDimensions have direct members defined in the same way as the InfoDimension through the IGOLAP Vocabulary, there is no difference between handling informational and topological dimension's members.

5.4.3.4 Dice

The dice operation delivers a composition of slice operations across multiple dimensions.

***Definition**: A dice operation delivers a new cube C', generated from a given cube C and selected members across two or more dimensions D. The emerging cube C' has the same schema as the initial cube C and the instances in C' are also instances of C.*

In this case, which is similar to the F_SLICE operation case, since there is no need to have hierarchical navigation through the cube, the aggregation is applied on the members of the observations in the given dataset. As was previously introduced, in the IGOLAP Vocabulary, the igolap:Member property is able to define both topological dimension members and informational dimension members. Through this introduction, the dice capability for the two types of dimension members was made available.

5.5 Summary

In this chapter the reason why current vocabularies cannot completely handle semantic data for OLAP were identified and an extended vocabulary was introduced. In addition the difference between informational and topological data, concepts that were briefly introduced in chapter 2- Research background, was investigated in more detail.

This chapter introduced the main OLAP operations that can be performed on the data modelled using the introduced vocabulary. In the following chapter it will be shown how a pipe of architectural OLAP operators can perform over the semantic web data modelled using the introduced vocabulary.

Chapter 6 – Materialisation of Integrated OLAP Operators for SW Databases

6.1 Introduction

It is mentioned in the previous chapter, Chapter 5 – IGOLAP Vocabulary Development, that the available vocabularies, including the chosen base vocabulary have a set of limitations in regards to:

- handling of both structures of data from the semantic web: topological and informational
- the capability to multi-dimensionally represent the semantic web data
- the supported construction of specialized OLAP operators

In Chapter 5, section OLAP operations of QB and QB4OLAP, was introduced that topological and informational data cannot be simultaneously handled by the OLAP operators introduced by the referenced vocabularies' implementations. Additionally, the possibility to use the OLAP operators over multiple databases to create further aggregations is another limitation of these approaches.

This chapter, based on the IGOLAP vocabulary previously introduced, shows how specialized OLAP operators can be implemented to address the above mentioned limitations.

This research refers to generic OLAP functionality, as such the related pros and cons in regards to ROLAP (Relational OLAP) and MOLAP (Multidimensional OLAP) or other types of traditional OLAP implementations are not considered in this approach. The MOLAP perspective in which multidimensional representation of the data is provided in the database, has no real impact on the efficient storage of the data as an additional dimension can be represented in a couple of more triples describing the same resource. The ROLAP perspective (in which updates can be easily generated) it can be achieved straightforwardly in semantic web data, due to the way in which semantic web technologies make it possible to publish, describe and reference resources on the web.

One aspect considered in this work is the possibility to materialise the generated views or observations. This can speed up the data querying and aggregation process required in any of the OLAP operators by giving the possibility to the operator implemented in this manner to make further aggregation based on the generated dataset from a previous used operator.

In order for this set of operators to be able to address the entire spectrum of information from semantic web, means means need to be provided to perform aggregations over both informational and topological data simultaneously, coming from multiple databases. Furthermore, this approach provides the possibility to implement all OLAP operations, including drill-down.

6.2 Architecture of Federated OLAP Operators (F_Operators)

The architecture introduced in this section, describes the architecture of the Federated OLAP Operators, based on the introduced IGOLAP Vocabulary from Chapter 5. Through this set of operators and vocabulary, the possibility of creating a middleware component is provided. This middleware, as presented through this work supports the definition and modelling of semantic web databases in preparation for performing OLAP operators on this data.

The challenge here is not to access multiple resources or semantic databases since these are access through the prefixing or mapping of the URI addresses, but to provided a multidimensional modelling vocabulary (IGOLAP) for these databases, and a set of operators based on these vocabulary (F_Operators) that can replicate the standard OLAP operators on any type of information modelled with the presented vocabulary.

The architecture of the Federated Operators (F_Operators), is based on the pipe-like design, in which the success of a preceding operator delivers the required input for the next operator. By orchestrating the F_Operators using this approach, a SELECT function can be replaced with another one, receiving other parameters, without changing the entire operator's orchestration. More precisely, in the architecture and implementation of F_Operators, this refers to the SELECT – CONSTRUCT sequence, in which a replacing the parameters of a SELECT doesn't change the behaviour of the CONSTRUCT function, but it will produce the corresponding output.

In the following subsections of this chapter there is a description of added capabilities and general OLAP characteristics included in the F_Operators' architecture.

6.2.1 General OLAP Characteristics

As previously introduced, a set of specialized OLAP operators (***Federated OLAP operators***) were designed to be able to operate simultaneously over multiple multidimensional semantic OLAP databases, merge the outputs into a common format and translate them accordingly to the desired output. Finally, this output can be ***materialised*** or only ***viewed***.

The set of operators has the role of a middleware, through which it provides access to semantic data in OLAP specific aggregations. As presented in Figure 6.3 Activity diagram of F_Operators' architecture, there are two inter-dependant flows that are separately initiated but rely on the same components to finish successfully. In consequence, the IGOLAP vocabulary is a key element in building the multidimensional semantic web databases on which OLAP aggregations can be performed. Secondly, the OLAP operators offer a set of validations and data retrieval options based on the concepts defined by the IGOLAP Vocabulary and instantiated in the databases' schemas. Furthermore, the set of OLAP operators act as a middleware between the user (person or application) and published semantic web databases, providing the means to create OLAP aggregations, visualise them and publish them if required.

The Federated OLAP operators need to interpret the requests according to a specific OLAP database in order to retrieve the data and convert it to a requested output-format. The Federated OLAP operators represent an extension of the classic OLAP operations as: roll-up, dice, drill down and slice.

Below are the definitions used in this work to guide the design and implementation requirements of the Federated OLAP Operators:

- A ***roll-up*** operation assumes a data summarization inside a given cube alongside a given dimension such as a given Cube C, a dimension D ∈ C and a dimension level lu∈ D, the Roll-up(C,D, lu) will return a new cube C' where measures are aggregated along D up to the level lu.
- ***Drill down*** is considered to be the reverse of the Roll up operation and assumes the disaggregation on a previously stored aggregation.

- ***Slice*** operation receives a cube C, a dimension D ∈ C and a member in level l ∈ D and returns a sub cube C', with the same schema except that all other member of the given level in dimension D are removed.
- In the ***dice*** operation a new cube C' is generated from a given cube C by selecting a set of members from two or more given dimensions. The emerging cube has the same schema as the initial cube C and the instances in C' are also instances of C.

As it emerges from the above definitions, the roll-up and drill-down operates by navigating through the hierarchy of a dimension. On the other hand, slice and dice operate on the members from the current hierarchical level of two or more dimensions, from a given cube.

The dimension operations that are used in this approach are referred throughout in this work as **F_OPERATORS**. These include the standard dimension operators as ***F_ROLL_UP, F_DICE, F_SLICE*** and ***F_DRILL***. These operators are derived from the standard OLAP dimension's operations, but are adapted to have the necessary functions to access multiple semantic web databases.

The standard OLAP measure operations are used as restriction functions in the dimension and include:

- AVG for retrieving the arithmetical mean of a set of numerical values
- SUM for the sum of a set of numerical values
- COUNT for the cardinality of a set of elements
- MIN and MAX for the minimum and maximum element of a set of elements.

As is mentioned throughout this work, agreeing with the generally accepted definition in the OLAP terminologies, the main two concepts applied in data warehousing are aggregate functions and OLAP operators. Although both represent some type of operations for data aggregation there exists a basic differentiation between the two as described below:

- ***Aggregate functions*** are applied on the measures of a level. As such hierarchical operators can make use of the aggregate function, but more than that, it assumes the inter-level aggregations.
- ***OLAP operators*** are used to retrieve or produce observations, navigating through the levels of a dimension. Aggregating the measures in a given selected level, assumes the usage of aggregated functions. Different OLAP operations define the combination and usage of different aggregation methods, as it results from the presented definitions of the operators.

As this is a quite straight-forward and easy to understand process for one dimension in a given cube containing only informational dimensions, applying OLAP operators becomes more complicated in the following scenarios:

- When the cube over which we want to apply this operator has also topological dimensions
- When the operators need to be performed over multiple databases and data cubes, in which case the dimensions hierarchy and dimensions types need to be handled in a way that is based on their described schema

Since there is available a set of aggregation options through SPARQL capabilities and given that this work delivers implementations of OLAP operators based on the SPARQL querying language, the focus of this work is on exemplifying the implementation of such operators based on the vocabulary introduced in Chapter 5 – IGOLAP Vocabulary Development.

6.2.2 Characteristics of Federated Operators

The ***F_OPERATORS*** include a set of processes defining the retrieval, building and normalisation stages as introduced below:

- If the datasets have different formats, data ***normalisation*** is performed before generating the output, this operation is included inside the F_Operators when the schema and data triplets are read.
- For the ***retrieval stage***, the operators identify the targeted databases, builds the SELECT operators for each database with given constraints, and gather

information from multiple databases by applying the built operators to specific datasets.

- In the ***building stage***, if the materialisation is required, the CONSTRUCT operator is initiated to compose the response from the retrieved data by applying the SELECT operator. If a materialisation of the created observations is not required, then generated output of the SELECT operator is displayed and released from memory. In this context a "pretty" display is used, in which the labels or "pretty name" for the levels and members are displayed instead of their identifiers.

In order to handle the data exchange, the F_OPERATORS are described as a pipe architecture containing a set of automatically built CONSTRUCT operator and a SELECT operator. If data normalisation is required before output is generated the third operator, the MERGE operator, is included in the F_OPERATOR pipe construction.

The MERGE operator is used to structure the partial RDF triple results from the SELECT operators using the same vocabulary for the output construction. Even though the MERGE concept has some similarity with the one in the semantic web pipes (Morbidoni, Polleres, Tummarello, & Le Phuoc, 2007), the MERGE from semantic web pipes is a simple join of the CONSTRUCT and/or SELECT operators output without normalisation capabilities and facilities to support Semantic OLAP Operators.

Figure 6.1 Integrated architecture of the introduced framework

Another difference in this approach is that, while CONSTRUCT and SELECT operators are a well identified block of code in the construction of F_Operators, the MERGE operator is mostly a tacit set of strategically placed methods designed to automatically address the needs of normalisation of F_Operators.

Given the above defined approach, the SELECT and CONSTRUCT operators are displayed for each F_OPERATOR and their implementation described for each of the F_Operators description.

Since F_OPERATORs are designed to access one or more than one OLAP database, they require a set of arguments in order to interpret the requests. Based on the arguments received, F_OPERATOR distinguish between:

- Single or multiple database access;
- Formatted or unformatted output;
- Request for view or request for materialisation of the output.

Some of the above identified functionality of the operators is not critical for the usage of the operator. This means that the parameters that can be provided to the call of the operators can be divided into two main categories:

- ***Mandatory parameters:*** these are parameters without which the operators cannot function: location of accessed SW OLAP implementation(s) (URIs or IRIs), dimensions (and dimension levels e.g. for F_ROLL_UP) and some others. These need to be validated before the aggregation is performed.
- ***Optional parameters***: these are parameters which if omitted the operator will give a standard output, so the operator's delivered functionality would not be changed by parameter value. For example the materialised or immaterialised output indication represents an optional parameter.

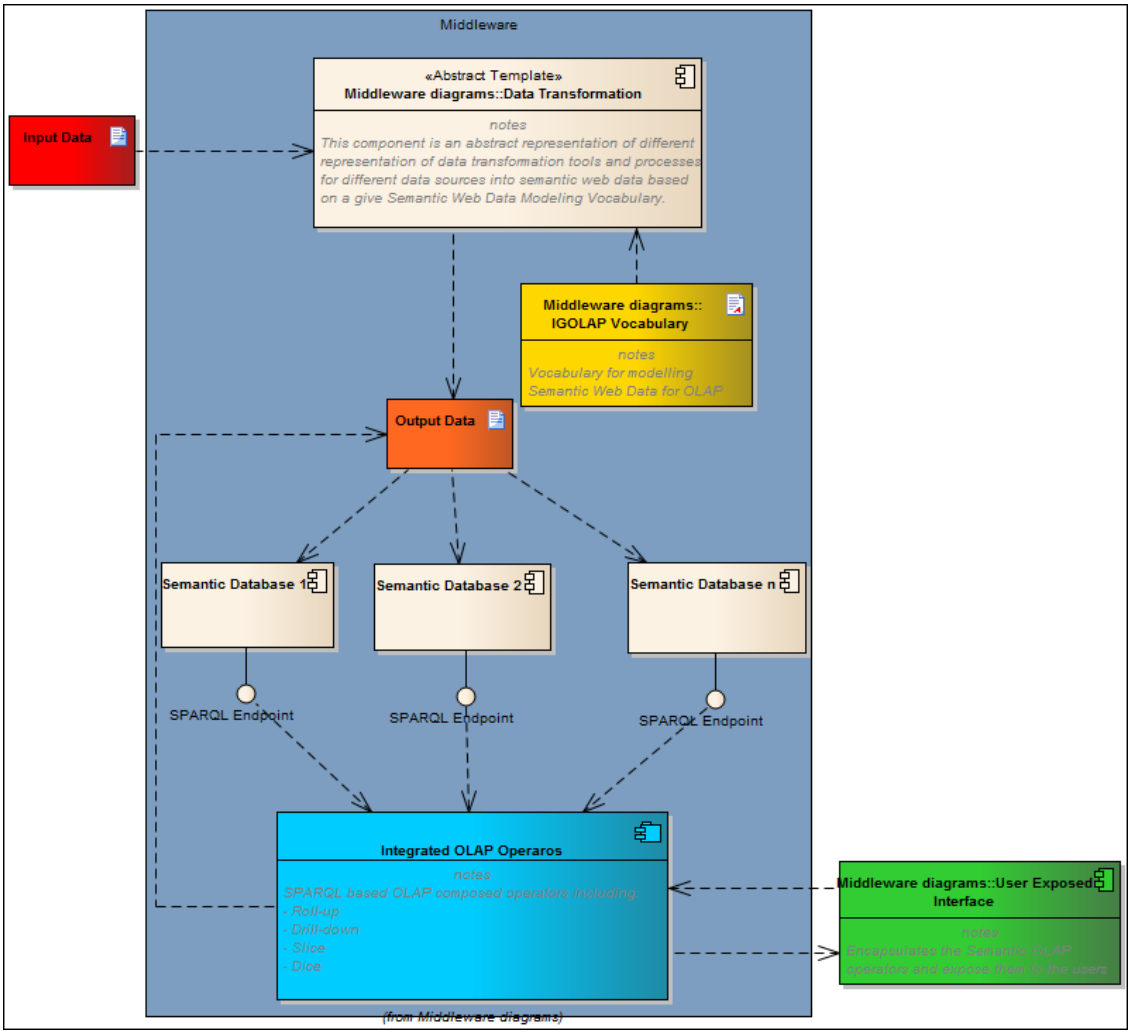


Figure 6.2 Integrated architecture for multiple databases access

The capability to operate over one or multiple databases is included in the implemented operators, by the way in which data is retrieved. The interface of the operator and the functionality delivered are encapsulated in one implementation.

A generic description of the architectural approach used to design the implementation of the presented F_Operators is presented in Figure 6.3.

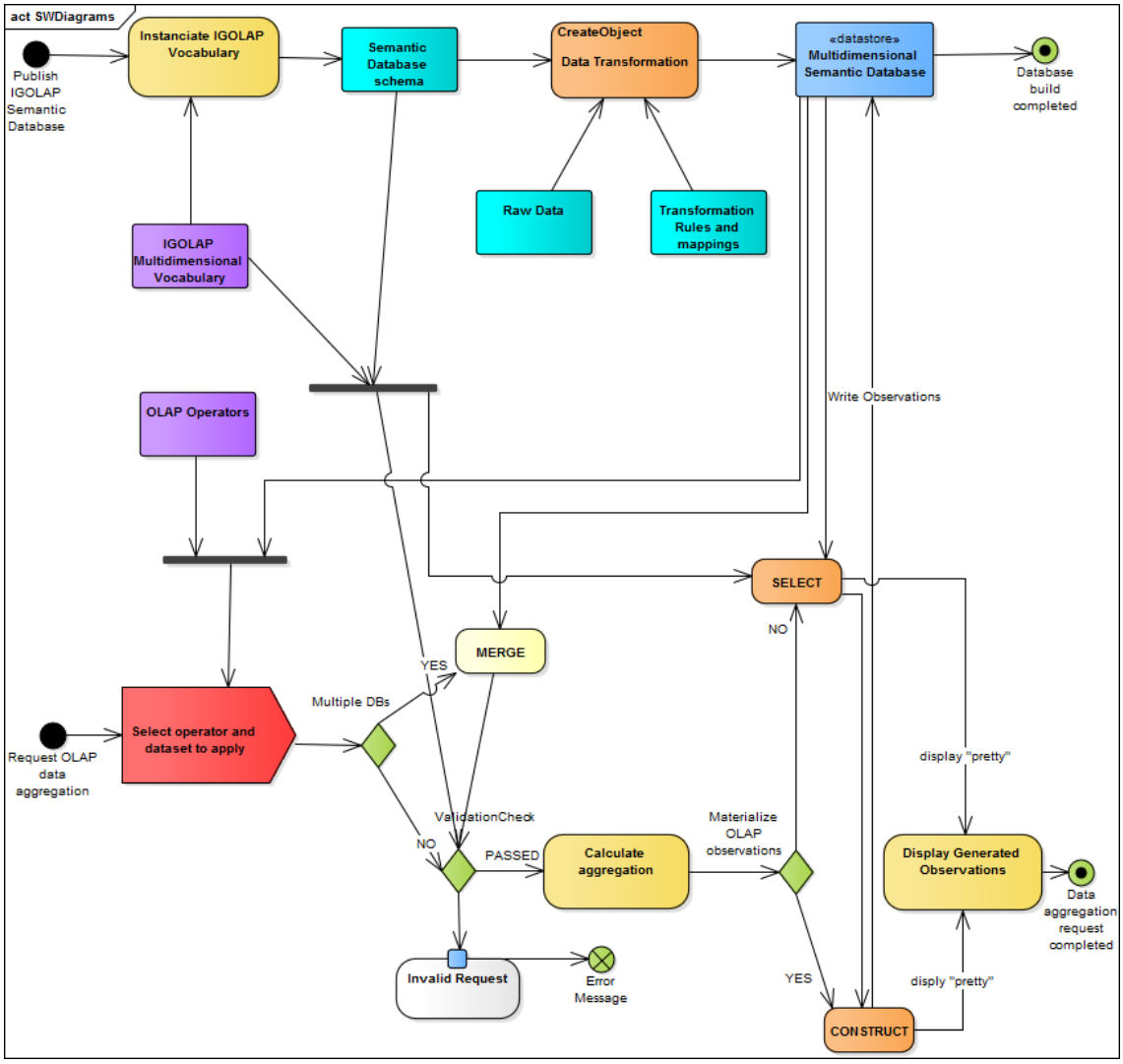


Figure 6.3 Activity diagram of F_Operators' architecture

Each one of the F_Operators defines a different type of aggregation as per the provided definitions in the first section of this chapter. The steps that define the basis of the development of all OLAP Operators introduced in the following subsections can be summarised as following:

- In **Step 1**, the request of any F_Operator is validated by verifying the number of parameters and their types as defined in the implementation of the possible constructors (generally there are two constructors available for each operator: one for single database access and one for multiple databases access)
- Each database implementation defines which dimensions are available in the implemented database and what the hierarchy levels look like. In **Step2**, the

concepts used to instantiate the dimensions, levels and particular constraints are identified, across one or multiple databases in order to be able to build the requested aggregation.

- In order to create the required aggregation, in the implementation of the F_Operators, **Step 3** checks if the measures required to be used are also valid measures across one or multiple databases and if the requested constraints can be applied on the measure's type.

  After the required measure and its constraints are validated, the observation is constructed, as defined in the implementation of each OLAP operator, based on a set of SELECT and CONSTRUCT pipe-like operators
- **Step 4** refers to the materialisation or visualisation option. If the materialisation option is selected, a set of observations can be published to be used in other aggregations. If this is not desired, it can just be displayed and released from the memory after the output was displayed.

Although the "materialisation" of a set of observations is an optional, in order to drill down and up through different levels of aggregations the observations representing the input data need to be stored in a database.

The Sequence Diagram introduced in

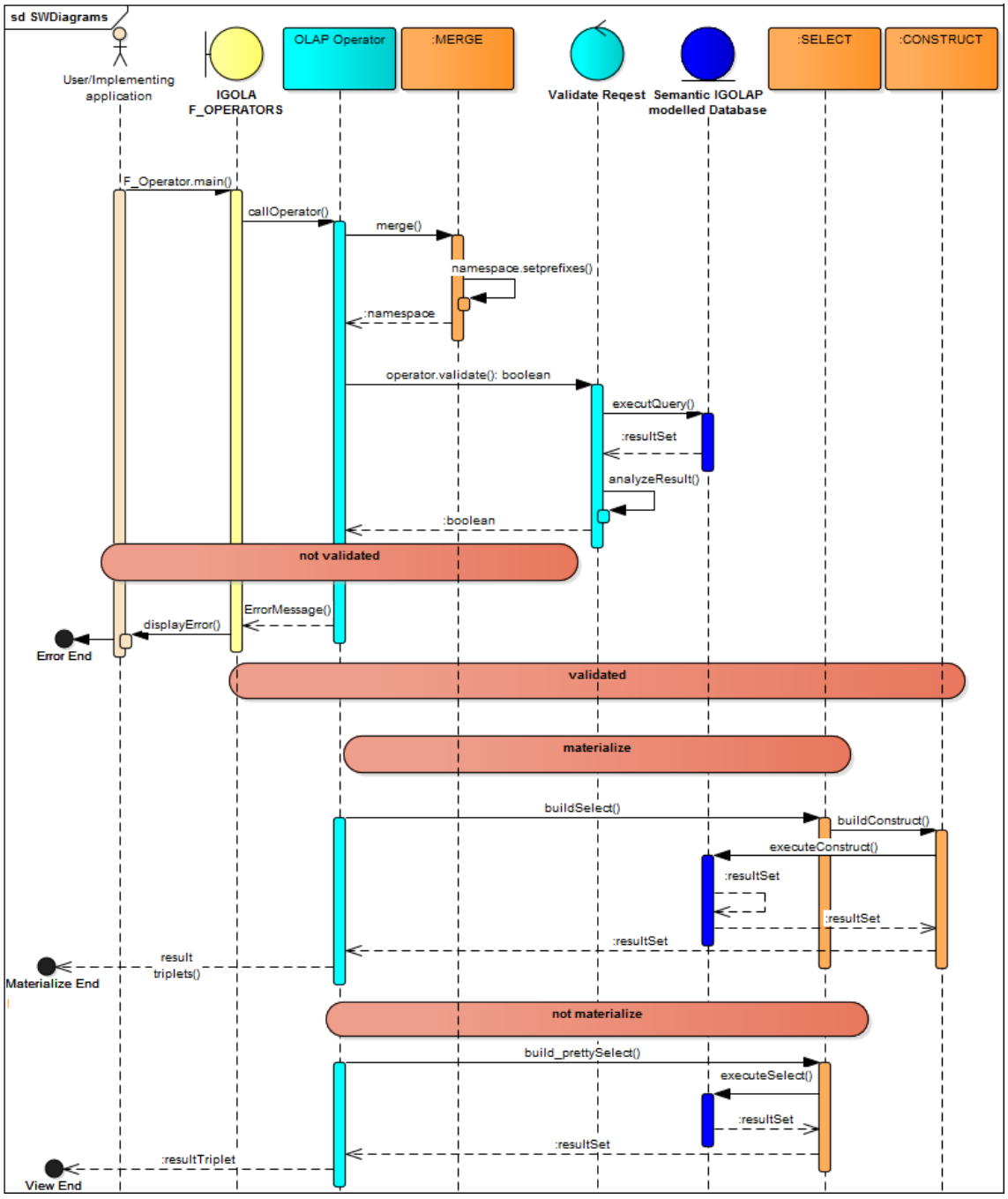


Figure 6.4 describes the base sequence on which all F_Operators are built. Depicted in this Diagram is the MERGE operator, introduced in the implementation section. The additional two operators SELECT and CONSTRUCT are introduced for each operators implementation. The diagram also describes the operators' behaviour based on the conditionals of continuation, which are the same for all operators: validity of the

request, materialisation or visualisation only request and the inter-calls of SELECT and CONSTRUCT operators.

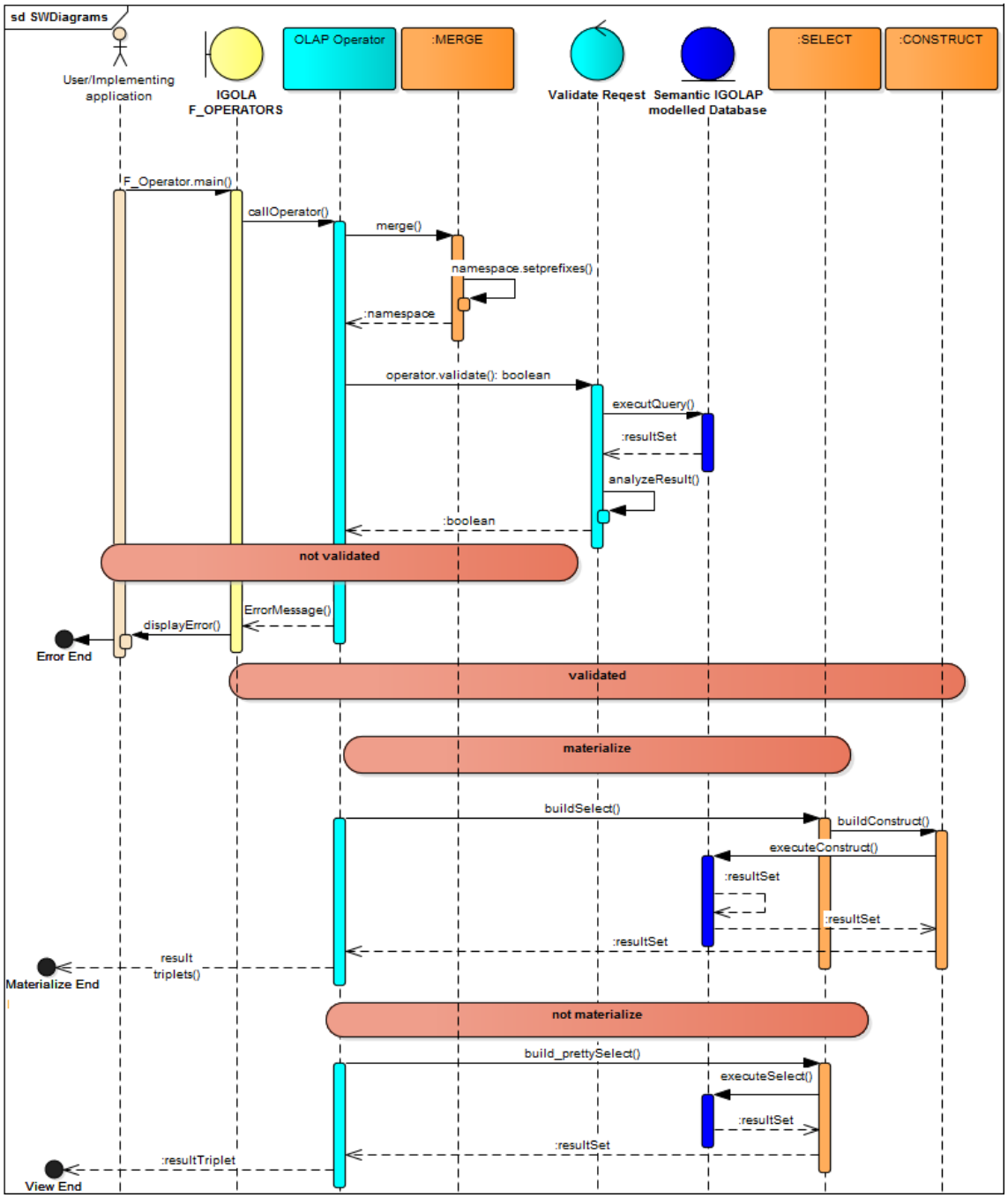


Figure 6.4 Sequence diagram of F_Operators

6.3 Implementation of F_Operators

The implementation of the above mentioned operators is realised in Java and makes use of Apache Jena libraries (The Apache Software Foundation, 2011).

The methods or steps that form the ***MERGE*** functionality and through which the integration of different database is delivered, are shared among all operators and are described below.

Step 1: First normalisation support; it is delivered through the possibility of loading various source data with a standard RDF/XML up to JVM memory. The data comes from different RDF serialisations, but they are normalised by employing the ModelFactory class for which sample code is shown below:

```
Model model = ModelFactory.createDefaultModel();

InputStream in = FileManager.get().open(dataset);
   if (in == null) {
      throw new IllegalArgumentException("File: " + dataset + " not found");
   }
   // read the N3 dataset file
   model.read(in, null, "N3");
```

Step 2: The prefixes or namespaces of the Vocabularies are located in the Namespaces class which gives the possibility of retrieving all the namespaces at run-time, but also to add the namespaces to the datasets and schema on which operators are applied. This gives the possibility of adding all database references to the same base namespace which operates at run-time:

```
package igolapOperators;

public class Namespace {
   static public final String NL = System.getProperty("line.separator");
   static private String prefix = "prefix dc: <http://purl.org/dc/elements/1.1/> " + NL
         + "prefix dcterms: <http://purl.org/dc/terms/> " + NL
         + "prefix igolap: <http://topublish.org/igolap#> " + NL
         + "prefix qb: <http://purl.org/linked-data/cube#> "+ NL
         + "prefix qb4o: <http://purl.org/olap#> " + NL
         + "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> " + NL
         + "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> " + NL
         + "prefix xml: <http://www.w3.org/XML/1998/namespace> "    + NL
         + "prefix xsd: <http://www.w3.org/2001/XMLSchema#> " + NL
         + "prefix ctr: <http://www.energydb.eu/igolap/Countries/> "        + NL
         + "prefix ct: <http://www.energydb.eu/igolap/Counties/> " + NL
         + "prefix c: <http://www.energydb.eu/igolap/Cities/> " + NL
         + "prefix pc: <http://www.energydb.eu/igolap/PostCodes/> " + NL
         + "prefix foaf: <http://xmlns.com/foaf/0.1/> " + NL
         + "prefix owl: <http://www.w3.org/2002/07/owl#> " + NL
         + "prefix scovo: <http://purl.org/NET/scovo#> " + NL
         + "prefix db1: <http://www.energydb.eu/igolap/schema#> " + NL
         + "prefix t: <http://www.energydb.eu/igolap/Data/Time#> " + NL
```

```
            + "prefix l: <http://www.energydb.eu/igolap/Data/Location#> " + NL
            + "prefix topo: <http://www.energydb.eu/igolap/Data/Topo#> " + NL
            + "prefix ds: <http://www.energydb.eu/igolap/Data/DataSets#>" + NL
            + "prefix dbpedia: <http://dbpedia.org/resource/> "+ NL
            + "prefix fn: <http://www.w3.org/TR/xpath-functions/> ";

    public static String getPrefixes() {
        return prefix;
    }

    public static void setPrefix(String pref, String uri) {
        prefix = prefix + NL + "prefix " + pref + ": <" + uri + "> ";
    }

}
```

Step 3: The validation and the actual aggregation calculation methods are constructed in a manner in which the name of the database to which the dimension, levels or members belong to is also required. The example below shows the simple method calls of the F_Roll_up operator for validating and constructing a single database:

```
public static boolean validate(String db_name, HashMap<String, String> aggLevelDim,
    String measure, String measureConstraint, Model …
```

```
public static void calculate(String db_name, Model schemasModel, Model dataModel, String
    measure, String measureConstraint, boolean materialize)
```

Step 4: The same serialization (N3) is also used in the materialisation process of the CONSTRUCT operator for consistency and better interoperability:

```
if (materialize) {
    String constructString = prefix + NL + "CONSTRUCT { "
        + "?id a qb:Observation; qb:dataSet
    ds:dataSet_"+rLevel+"_"+measure+"_"+measureConstraint+" ; "+db_name+":"+rLevel+"
    ?"+rLevel+"; "+q3a+" "+db_name+":"+measure+" ?"+measure+"}"
        + "WHERE{"
        + "       {"
        + SelectConstruct
        + "}"
        + "}";

        Query constructQuery = QueryFactory.create(constructString);
        QueryExecution constr = QueryExecutionFactory.create(constructQuery, m2);
        Model obsModel = constr.execConstruct();
        obsModel.write(System.out);
        OutputStream output = new FileOutputStream(dataWrite_path);
        obsModel.write(output, "N3", null);
```

The CONSTRUCT and SELECT operators are specific to each F_Operator implementation and presented separately in the following subsection. The automated way in which they are build is also described and their application in concrete

aggregation queries is also presented alongside the generated outputs. A set of features and/or clauses specific to SELECT and CONSTRUCT operators and respectively for the OLAP F_Operators are also introduced.

Due to increasing complexity or weight of the SPARQL queries, through usage of filters and complex SPARQL operators, the presented design includes only a limited number of SPARQL operators and complex functions. This is illustrated in Table 6.1:

	F_Roll_up		F_Drill_down		F_Slice		F_Dice	
	View	materialize	view	materialize	view	materialize	view	materialize
SELECT	x	x	x	x	x	x	x	x
CONSTRUCT		x		x		x		x
GROUP	x	x	x	x	x	x	x	x
DISTINCT	x	x	x	x	x	x	x	x
BIND		x		x		x		x
VALUES							x	x
IRI		x		x		x		x
SUBSTR		x		x		x		x

Table 6.1 SPARQL 1.1. included operators and functions in F_Operators implementation

The BIND, IRI and SUBSTR functions are used to support the SPARQL CONSTRUCT operator. Since these functions are used to build the needed identifiers to be used in delivering the required graph section. These are not needed in the case of the "view-only" option of the F_Operators, since in these cases the aggregated values are only displayed and not added to the graph. Additionally the VALUES function supports the selection and usage of specific values, in the case of F_Dice the multiple needed dimension and members.

The definition of the dimensions, levels, members and measures for DB1, which was introduced and used throughout this work, is introduced below. These definitions are used for all exemplifications of the operators presented in this chapter. The whole set of code or data used for the exemplification of this section can be found in Appendix 2.1. of this work.

Four dimensions are used in the following examples: Time, Location, Household and Income.

```
#-Dimensions definitions of DB1 based on
   IGOLAP Vocabulary

db1:Location a igolap:InfoDimension .

db1:Time a igolap:InfoDimension ;
    rdfs:label "All time "@en.

db1:Household a igolap:TopoDimension;
    igolap:topoDConnectedTo db1:Income .

db1:Income a igolap:TopoDimension ;
    igolap:topoDConnectedTo
   db1:Appliance,
        db1:Household .
```

```
#--- Sample Levels definitions of DB1,
   for InfoDimensions Location and Time

db1:Country a qb4o:LevelProperty ;
    qb4o:inDimension db1:Location ;
    igolap:childLevel db1:Region .
db1:County a qb4o:LevelProperty ;
    qb4o:inDimension db1:Location ;
    qb4o:parentLevel db1:Region ;
    igolap:childLevel db1:City .

[..]
db1:Year a qb4o:LevelProperty,xsd:gYear;
    qb4o:inDimension db1:Time ;
    igolap:childLevel db1:Month .

db1:Month a qb4o:LevelProperty,
   xsd:gYearMonth;
   qb4o:inDimension db1:Time ;
   qb4o:parentLevel db1:Year ;
         igolap:childLevel db1:Day .
```

Figure 6.5 Dimensions and Levels definitions

```
topo:hh106 a igolap:Member ;
    rdfs:label "106"@en ;
    igolap:ofDimension db1:Household ;
    igolap:topoDConnectedTo topo:ukIrang3.
 [..]
topo:ukIrang2 a igolap:Member ;
    rdfs:label "Income range between 10000 and 20000"@en ;
    igolap:ofDimension db1:Income ;
    db1:hasMaxIncome "20000"^^xsd:integer;
    db1:hasMinIncome "10000"^^xsd:integer.
```

Figure 6.6 Sample members of TopoDimension household and income

The definitions of these dimensions are shown in (Figure 6.5) and a few sample members across both topological dimensions and informational dimensions' members are also available in (Figure 6.6) (Figure 6.7) respectively.

```
 # --- Sample members for the levels of dimension Location ---

l:uk a igolap:Member ;
    rdfs:label "United Kingdom"@en ;
    qb4o:inLevel db1:Country ;
    igolap:childLevel l:rWestMid ;
      igolap:childLevel l:rSouthWest ;
      igolap:childLevel l:rNorthEast .

l:rWestMid a igolap:Member;
      rdfs:label "West Midlands Region"@en ;
    qb4o:inLevel db1:Region;
    igolap:childLevel l:cWestMid ;
    qb4o:parentLevel l:uk .

t:D01M01Y2011 a igolap:Member ;
 rdfs:label "1st of January 2011@en";
```

Figure 6.7 Sample members of levels in InfoDImensions

The datasets on which the operators applied, can be aggregated to deliver a specific information from a detailed level, but it required a certain operator to operate on. As a consequence, the datasets used for each operator are described in each respective exemplification subsection and available in the Appendix B.

6.3.1 F_Roll_up

In order to exemplify the implementation and operation of the F_Roll_up operator, a walk-through of the implementation and exemplification through queries is provided in the following paragraphs. In order to better follow the example and the correctness of the operator's functionality, only a small sample of the dataset from the previously introduced DB1 is used.

The entire data samples, including namespaces, are provided in the Appendix B. For better readability, the namespaces are generally also omitted in this chapter.

6.3.1.1 Implementation

This section is designed as a walk-through the implementation of the F_Roll_Up operator. The code presented below is the java code of the operator's implementation.

Due to the fact that MERGE functionality is distributed throughout the implementation of the operators and it is introduced in the previous section, in this subsection the focus is on detailing the construction of the remaining building blocks of the operator.

As presented in the sequence diagram introduced previously (Figure 6.4), the remaining components include the validation and construction of the observations based on SELECT and CONSTRUCT operators based on the materialisation or only viewing constraints. These components were used in the implementation of the two core methods of this operator: validate and calculate. As a consequence the definition of the F_Roll_up operator is implemented as follows:

```
public Roll_up(String dbSchemas_folder_path, String dataset_path, String db_name,
        HashMap<String, String> aggLevelDim, String measure, String
        measureConstraint, boolean materialize, String materialize_path)
{
   Model schemasModel = ModelFactory.createDefaultModel();
   Model dataModel = ModelFactory.createDefaultModel();
```

```
      if (materialize){
         if (materialize_path!=null){
            dataWrite_path = materialize_path;
         } else {
            dataWrite_path = dbSchemas_folder_path+"\\Dataset_new.ttl";
         }
      }
      schemasModel = load_db(dbSchemas_folder_path, schemasModel);
      dataModel = load_dataSet(dataset_path, dataModel);
      if (validate(db_name, aggLevelDim, measure, measureConstraint,
            schemasModel, dataModel)){
         calculate(db_name, schemasModel, dataModel, measure,
                  measureConstraint, materialize);
      }
      schemasModel.close();
      dataModel.close();
   }
```

Two main aspects of the operator's design are retrieved in the constructor's declaration: on the one side, the database schema and the dataset model are computed initially separately to reduce the validation time through executing the queries only on the right subset of the triplets. On the other hand, there is no differentiation between the topological and informational dimensions' members in the constructor construction. This is due to the fact that this information is retrieved in the validation process passed as a *private parameter* of the operator to the calculation of the aggregation.

The ***validate()*** method makes sure the requested OLAP operation is applicable to the give schemas and datasets in terms of syntax and grammar.

```
public static boolean validate(String db_name,
         HashMap<String, String> aggLevelDim, String measure,
         String measureConstraint, Model schemasModel, Model dataModel) {

   Iterator<String> it = aggLevelDim.keySet().iterator();
   if (it.hasNext()) {
      rLevel = it.next();
      rDim = aggLevelDim.get(rLevel);
      try {
      // find the type of the dimension to be aggregated over (Info or Topo) and
      //retrieve the required levels or members on which needs to be operated
      String queryDimensionType = prefix + NL
               + "SELECT ?dimensionType WHERE {" + db_name + ":"
               + rDim + " a ?dimensionType .}";
      Query queryDimension = QueryFactory.create(queryDimensionType);
      QueryExecution qExe = QueryExecutionFactory.create(queryDimension,schemasModel);
      ResultSet dimType = qExe.execSelect();
      QuerySolution dType = dimType.next();
      dimensionType = dType.get("dimensionType").toString();

      if (dimensionType.equalsIgnoreCase(Vocabulary.igolap+ "InfoDimension")) {
         String queryString = prefix + NL
               + "SELECT ?childLevel WHERE {" + db_name + ":"+ rLevel
               + " a qb4o:LevelProperty; qb4o:inDimension " + db_name + ":" + rDim
               + ";igolap:childLevel ?childLevel .}";
         Query query = QueryFactory.create(queryString);
         QueryExecution qe = QueryExecutionFactory.create(query,    schemasModel);
         ResultSet results = qe.execSelect();
         while (results.hasNext()) {
            QuerySolution qs = results.next();
            levelChildren.add(qs.get("childLevel").toString());
            }
         if (levelChildren.isEmpty() || levelChildren == null) {
```

```
            System.out.println("The requiered level doesn't have child levels!");
            return false;
         }
      } else {
         if (dimensionType.equalsIgnoreCase(Vocabulary.igolap+ "TopoDimension")) {
            String queryString = prefix+ NL+ "SELECT ?topoConnected WHERE { "
                      + "?topoConnected a igolap:TopoDimension; igolap:topoDConnectedTo"
                      + db_name + ":" + rDim + " .}";
            Query query = QueryFactory.create(queryString);
            QueryExecution qe = QueryExecutionFactory.create(query, schemasModel);
            ResultSet results = qe.execSelect();
            boolean areConnected = false;
            while (results.hasNext()) {
              QuerySolution qs = results.next();
              String c=qs.get("topoConnected").toString().replaceAll(Vocabulary.db1,"");
              if (c.equalsIgnoreCase(rLevel)) {areConnected = true;}
            }
            if (areConnected==false) {return areConnected;}
         }
      }
      // validate that child and measure are in the datasets
      String selectTopoString = prefix + NL + "SELECT ?TopoDims "
            + "WHERE {?ds a qb:DataStructureDefinition;"
            + "?p ?o . ?o igolap:TopoDimension ?TopoDims" + " }";
      Query selectTopoComp = QueryFactory.create(selectTopoString);
      QueryExecution retrieveTopoComp = QueryExecutionFactory.create(selectTopoComp,
   dataModel);
      ResultSet topoRes = retrieveTopoComp.execSelect();
      while (topoRes.hasNext()) {
         QuerySolution qs = topoRes.next();
         topos.add(qs.get("TopoDims").toString());
      }
      String selectInfoLevelString = prefix + NL
            + "SELECT ?InfoLvls WHERE {?ds a qb:DataStructureDefinition;"
            + "?p ?o . ?o qb4o:level ?InfoLvls" + " }";
      Query selectInfoComp = QueryFactory.create(selectInfoLevelString);
      QueryExecution retrieveInfoComp = QueryExecutionFactory.create(selectInfoComp,
   dataModel);
      ResultSet infoRes = retrieveInfoComp.execSelect();
      while (infoRes.hasNext()) {
         QuerySolution qs = infoRes.next();
         infoLevels.add(qs.get("InfoLvls").toString());}

      if (dimensionType.equalsIgnoreCase(Vocabulary.igolap + "InfoDimension")) {
         for (int i = 0; i < infoLevels.size(); i++) {
            for (int j = 0; j < levelChildren.size(); j++) {
               if (infoLevels.get(i).equalsIgnoreCase(levelChildren.get(j))) {
                     rIMember = true;}
            }
         }
      } else if (dimensionType.equalsIgnoreCase(Vocabulary.igolap   + "TopoDimension")) {
         for (int i = 0; i < topos.size(); i++) {
               if (topos.get(i).equalsIgnoreCase(Vocabulary.db1 + rDim)) {
               rTMember = true; }
         }
      }

      String selectMeasureString = prefix + NL + "SELECT ?measure "
            + "WHERE {?ds a qb:DataStructureDefinition;"
            + "?p ?o . ?o qb4o:measure ?measure }";
      Query selectMeasureComp = QueryFactory.create(selectMeasureString);
      QueryExecution retrieveMeasureComp =
      QueryExecutionFactory.create(selectMeasureComp, dataModel);
      ResultSet measureRes = retrieveMeasureComp.execSelect();

      boolean measure_valid = false;
      while (measureRes.hasNext()) {
         QuerySolution qs = measureRes.next();
         String found_measure = qs.get("measure").toString();
         measures.add(found_measure);
         if (found_measure.equalsIgnoreCase(Vocabulary.db1 + measure)) {
            measure_valid = true;
         }
```

```
      }
      String measureConsString = prefix + NL + "SELECT ?measureAgg "
            + "WHERE {?measureAgg a qb4o:AggregateFunction }";
      Query selectMeasureCostr = QueryFactory.create(measureConsString);
      QueryExecution retrieveMeasureCostr =
      QueryExecutionFactory.create(selectMeasureCostr, schemasModel);
      ResultSet constrains = retrieveMeasureCostr.execSelect();

      boolean exist_constrain = false;
      while (constrains.hasNext()) {
         QuerySolution qs = constrains.next();
         String found_measure = qs.get("measureAgg").toString();
         if (found_measure.equalsIgnoreCase(Vocabulary.qb4o
               + measureConstraint)) {
            exist_constrain = true; }
      }
      if ((measure_valid)&&(exist_constrain)&&((rIMember)||(rTMember))){return true;
      } else { return false;  }
   } catch (Exception e) {
      e.printStackTrace();
      System.out.println("Dimensions retrieval exception!");
      return false;
   }
} else {
   System.out.println("No given roll-up target!");
   return false;
}}
```

The measure constraint reflects one of the aggregations functions that the Vocabulary has to offer, as such in the validation process it is also verified that the requested aggregation function is a valid one according to the IGOLAP Vocabulary.

After ***validate()*** method was run, if all the validation conditions were met, the ***calculate()*** method is called and the following private parameters of the operator are instantiated:

```
   static private List<String> levelChildren=new ArrayList<String>();//sublevel required
   static private List<String> topos=new ArrayList<String>();//TopoDimensions in dataset
   static private List<String> infoLevels = new ArrayList<String>();//InfoDimensions
   static private String dimensionType;//Info/TopoDim. aggregated dimension requested
```

The calculate method builds automatically the SELECT and the CONSTRUCT request based on two constraints (or input and calculated parameters):

- It depends on the requested dimension over which the aggregation is performed. It is either topological or informational.
- If there is a request for the generated observations to be written to the database.

The difference between topological and informational type of dimension does not affect the template of the SELECT and CONSTRUCT queries, just the way in which the aggregation relationship is passed to the operators. As such, below the building

process of the informational SELECT query and the required changes for the topological dimensions SELECT is presented.

This is followed by the CONSTRUCT for the materialisation and viewing if no materialisation required. In the case of this operator *no changes are required for the topological dimensions*.

The changes applied for the topological dimensions in the SELECT operator are highlighted after the presented coded excerpt:

```
// SELECT
if (dimensionType.equalsIgnoreCase(Vocabulary.igolap + "InfoDimension")) {
   [..]
   Model m2 = ModelFactory.createUnion(schemasModel, dataModel);
   for (int i = 0; i < infoLevels.size(); i++) {
         iLevels.add(infoLevels.get(i).replaceAll(Vocabulary.db1, ""));
         if (!(infoLevels.get(i).equalsIgnoreCase(levelChildren.get(0)))) {
            q1 = q1 + " ?" + iLevels.get(i);
            q3a = q3a+db_name+ ":" + iLevels.get(i) + " ?"+ iLevels.get(i) + ";";
            q1Visual=q1Vis+ " ?" + iLevels.get(i) + "_label";
            q4Visual=q4Vis+"?"+iLevels.get(i)+"rdfs:label ?"+iLevels.get(i)+"_label.";
         }
      }
      for (int i = 0; i < topos.size(); i++) {
         tDims.add(topos.get(i).replaceAll(Vocabulary.db1, ""));
         q1 = q1 + " ?" + tDims.get(i);
         q3a = q3a + db_name + ":" + tDims.get(i) + " ?"+ tDims.get(i) + ";";
         q1Visual = q1Visual + " ?" + tDims.get(i) + "_label";
         q4Visual = q4Visual+"?"+tDims.get(i)+" rdfs:label ?"+tDims.get(i) + "_label .";
      }
      String q1a = " (" + measureConstraint + "(?measure) AS ?"+ measure + ") ";
      String q3b = q3a + db_name + ":" + measure + " ?measure .";
      String q0 = "SELECT DISTINCT ";
      String q0Constr = q0 + "?" + rLevel + " ?id";
      String q0Visual = q0 + "?" + rLevel + "_label";
      String q2 = " WHERE {" + "?o a qb:Observation; ";
      String q3 = "<" + levelChildren.get(0) + "> ?child; ";
      q4 = "?child qb4o:parentLevel ?" + rLevel + ".";
      q4Visual = q4Visual + "?" + rLevel + " rdfs:label?" + rLevel+ "_label .";
      String q5 = "} GROUP BY ";
      String q5Constr = q5 + "?" + rLevel + " ?id" + q1;
      String q5Visual = q5 + "?" + rLevel + "_label" + q1Visual;
      String q4Constr = "BIND (iri(concat(\"" + Vocabulary.ds+ "\",STRAFTER(str(?"+
   tDims.get(0)+ "), \"#\"),\"_\",STRAFTER(str(?" + rLevel  + "),\"#\"))) AS ?id)";

      String SelectConstruct = q0Constr + q1+q1a+q2+q3+q3b+q4+q4Constr + q5Constr;
      String SelectVisualize = q0Visual + q1Visual+q1a+q2+q3+q3b+q4+q4Visual+ q5Visual;
```

The only change in the SELECT operator for the topological dimension is highlighted above and presented below:

```
if (dimensionType.equalsIgnoreCase(Vocabulary.igolap+ "TopoDimension")) {
[..]
String q3 = db_name + ":" + rDim + " ?topoM;";
q4 = "?topoM igolap:topoDConnectedTo ?" + rLevel + ".";
```

This is made possible due to the extensions to the Vocabulary added by the IGOLAP vocabulary as presented in Chapter 5.

If the materialisation is required, the CONSTRUCT operator is dynamically built as follows:

```
// CONSTRUCT
String constructString = prefix+NL+"CONSTRUCT {?id a qb:Observation; qb:dataset"
      +ds:dataSet_"+rLevel+"_"+measure+ "_"+ measureConstraint+ " ; "+db_name+ ":"
      +rLevel+" ?"+rLevel+"; "+q3a+" "+db_name+ ":" + measure + " ?"+ measure + "}"
      + "WHERE{" + " {"
      + SelectConstruct + "}" + "}";
[..]
```

Additionally to writing the construct query automatically for OLAP operators, the schema of the newly developed operators is in this case also automatically generated at run-time.

```
// CONSTRUCT observations new data structure and dataset definition
String s = "ds:DailyHhECons a qb:DataStructureDefinition;" + NL
      + "qb:Component [igolap:TopoDimension "+ db_name + ":" + rLevel + "];" + NL;
         for (int j = 0; j < iLevels.size(); j++) {
            s = s + "qb:Component [qb4o:level "+db_name+":"+ iLevels.get(j) + "];" + NL;
         }
         for (int j = 0; j < tDims.size(); j++) {
            if (!(tDims.get(j).equalsIgnoreCase(rDim))) {
            s=s+"qb:Component [igolap:TopoDimension "+db_name+":"+tDims.get(j)+"];"+ NL;
            }
         }
         s = s + "qb:Component [qb4o:measure " + db_name + ":"+ measure + ";" + NL
            + "qb4o:hasAggregateFunction qb4o:"+ measureConstraint + "]." + NL;
         String datasetSchema = s + "ds:dataSet_" + rLevel + "_"+ measure + "_"
            +measureConstraint+ " a qb:DataSet;" + NL+ " qb:structure ds:DailyHhECons.";
         output.write(datasetSchema.getBytes(Charset.forName("UTF-8")));
```

The call of the F_Roll_up operator is identical in this case for aggregating over both topological and informational.

```
   public static void main(String[] args) {
      HashMap<String, String> aggLevelDim = new HashMap<String, String>();
      aggLevelDim.put("Month", "Time");

      HashMap<String, String> aggTopoDims = new HashMap<String, String>();
      aggTopoDims.put("Income", "Household");

      boolean materialize = true;
      String measureConstraint="AVG";

      String dbSchemas_path= "… schemas\\";
      String dataset_path="… DB1\\data\\Dataset_Day_Cons.ttl";
      String dataWrite_path_s="… DB1\\data\\Dataset_Month_Avg.ttl";

      String measure = "eCons";
      // can be use either aggLevelDim parameter for InfoDimensions or aggTopoDims for
   TopoDimensions
      new Roll_up(dbSchemas_path,dataset_path,"db1", aggLevelDim, measure,
   measureConstraint, materialize, dataWrite_path_s);
   }
```

The generation of the SPARQL queries, through the execution of the above code for the SELECT and CONSTRUCT, are presented for specific attributes, as required in the queries introduced in the next section.

6.3.1.2 Exemplification

This subsection introduces a set of queries that require rolling up aggregations on both topological and informational dimensions. The F_Roll_up operator is used to deliver these aggregations on the given schemas of the datasets and databases, and the definition of the members, which was described previously. This section also provides example use of the ***validate()*** and the ***calculate()*** methods.

One observation structure from the dataset used is presented below and the entire data used for this exemplification can be found in Appendix B2.2.

```
ds:o1 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D08M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "4.01"^^xsd:decimal .
```

The ***validate()*** method can be summarised in six Steps as follows:

```
Step 1.retrieve dimensionType of the dimension to which the required level belongs.

Step2.
   If dimensionType equals "InfoDimension" then {
         Retrieve childLevels of required Level;
   }
   If dimensionType equals "TopoDimension" then{
      Validate if the target TopoDimension is "topoConnected" to the start TopoDimension
   }
Step 3.
   For all (dimension in dataset) {
      Find if chidlLevels or dimension is one of them;}
Step 4.
   For all (measure in dataset){
      Find if required dimension is one of them;}
Step 5.
   For all (AggregatedFunctions defined in IGOLAP){
      Find if measureConstraint is one of them;}
Step 6. If Step1 to 5 true then validation returns true.
```

Query 1: Monthly energy consumption of a dataset (cube) containing daily consumption measurements.

In this query, a roll-up is used to aggregate data from the granularity level "Day" to the level "Month" with the summing aggregation function.

For this the roll-up function call takes the following values for the relevant parameters:

```
aggLevelDim: "Month", "Time"
measure: "eCons"
measureConstraint: "SUM"
```

In this query example the validation method delivers:

```
Step 1.retrieve dimensionType of dimension "Time".
   Returns: InfoDimension

Step2.
   If "InfoDimension" equals "InfoDimension" then {
         Retrieve childLevels of required Level;
   }
   Returns: childLevels ={"Day"}
   If "InfoDimension" equals "TopoDimension";
Returns: true

Step 3.
   For all (dimension or levels in dataset) {
      If "Day" in {Day, City, Household}}
Returns: true

Step 4.
   For all (measure in dataset){
      If "eCons" in {"eCons"}}
Returns: true

Step 5.
   For all (AggregatedFunctions defined in IGOLAP){
      If "SUM" in {"SUM", "AVG", "COUNT", "MIN", "MAX"};}
Returns: true

Step 6. If Step 2 to 5 are true then validation returns true.
```

Since the ***validate()*** method presented above passed, the second method, ***calculate()*** is called. For this query, the ***calculate()*** method builds the SELECT and/or CONSTRUCT queries as illustrated below:

```
SELECT DISTINCT  ?Month_label
   ?City_label ?Household_label
   (SUM(?measure) AS ?eCons)
  WHERE  {
   ?o rdf:type qb:Observation .
   ?o db1:Day ?child .
   ?o db1:City ?City .
   ?o db1:Household ?Household .
   ?o db1:eCons ?measure .
   ?child qb4o:parentLevel ?Month
   .
   ?City rdfs:label ?City_label .
   ?Household rdfs:label
   ?Household_label .
   ?Month rdfs:label ?Month_label
    }
  GROUP BY ?Month_label
?City_label ?Household_label
```

```
CONSTRUCT
{?id rdf:type qb:Observation .
 ?id qb:dataSet ds:dataSet_Month_eCons_SUM .
 ?id db1:Month ?Month .
 ?id db1:City ?City .
 ?id db1:Household ?Household.
 ?id db1:eCons ?eCons .}
 WHERE {{  SELECT DISTINCT ?Month ?id ?City
  ?Household (SUM(?measure) AS ?eCons)
    WHERE {?o rdf:type qb:Observation .
     ?o db1:Day ?child .
     ?o db1:City ?City .
     ?o db1:Household ?Household .
     ?o db1:eCons ?measure .
     ?child qb4o:parentLevel ?Month
  BIND(iri(concat("http://www.energydb.eu/igola
  p/Data/DataSets#",strafter(str(?Household),
  "#"),"_",strafter(str(?Month), "#"))) AS ?id)
    }
GROUP BY ?Month ?id ?City?Household
   }}
```

The output of the function is provided below for both values of the parameter ***materialize***:

```
materialize = false;
-------------------------------------------------------------------------
| Month label        | City label      | Household label | eCons  |
=========================================================================
| "March 2011@en"    | "Birmingham"@en | "106"@en        | 45.80  |
| "March 2011@en"    | "Birmingham"@en | "2"@en          | 185.90 |
| "February 2011@en" | "Birmingham"@en | "106"@en        | 140.02 |
| "February 2011@en" | "Birmingham"@en | "119"@en        | 395.63 |
| "February 2011@en" | "Birmingham"@en | "154"@en        | 185.50 |
| "February 2011@en" | "Birmingham"@en | "2"@en          | 231.07 |
| "April 2011@en"    | "Birmingham"@en | "119"@en        | 31.30  |
-------------------------------------------------------------------------
```

```
materialize = true;
ds:DailyHhECons a qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension db1:Household];
qb:Component [qb4o:measure db1:eCons;
qb4o:hasAggregateFunction qb4o:SUM].

ds:dataSet Month eCons SUM a qb:DataSet;
 qb:structure ds:DailyHhECons.

ds:hh119_M02Y2011  a   qb:Observation ;
        qb:dataSet     ds:dataSet_Month_eCons_SUM ;
        db1:City       l:birm ;
        db1:Household  topo:hh119 ;
        db1:Month      t:M02Y2011 ;
        db1:eCons      395.63 .

ds:hh2 M02Y2011  a     qb:Observation ;
        qb:dataSet     ds:dataSet Month eCons SUM ;
        db1:City       l:birm ;
        db1:Household  topo:hh2 ;
        db1:Month      t:M02Y2011 ;
        db1:eCons      231.07 .

ds:hh154 M02Y2011  a   qb:Observation ;
        qb:dataSet     ds:dataSet Month eCons SUM ;
        db1:City       l:birm ;
        db1:Household  topo:hh154 ;
        db1:Month      t:M02Y2011 ;
        db1:eCons      185.50 .

ds:hh2_M03Y2011  a     qb:Observation ;
        qb:dataSet     ds:dataSet_Month_eCons_SUM ;
        db1:City       l:birm ;
        db1:Household  topo:hh2 ;
        db1:Month      t:M03Y2011 ;
        db1:eCons      185.90 .
[..]
```

Query 2: Average daily consumption based on households' income range.

This query requires a roll_up aggregation across topological dimensions to generate the desired outcomes. In the case of topological dimension, these do not have hierarchy levels and as such the aggregation happens based on the dimensions' connection relationship to another topological dimension.

Income is a ***topological dimension*** in this database, but its members have no measure relating to energy consumption. Additionally the Household topological dimension has members which are connected with members of the topological dimension Income. As such, it is desired to perform a roll up aggregation in which members of the Household dimension are grouped based on their relationship with the same Income dimension members. The output should be the different Income ranges and the corresponding average energy consumption for that range.

For this query, the roll up function call takes the following values for the relevant parameters:

```
aggLevelDim: "Income", "Household"
measure: "eCons"
measureConstraint: "AVG"
```

These parameters are passed to the ***validate()*** method which performs the validation process. This process is illustrated below.

```
Step 1.retrieve dimensionType of dimension "Household".
Returns: TopoDimension

Step2.
   If "TopoDimension" equals "InfoDimension";
   If "TopoDimension" equals "TopoDimension" then {
         if "Household igolap:topoDConnectedTo Income";
   }
Returns: true

Step 3.
   For all (dimension or levels in dataset) {
      If "Household" in {Day, City, Household}}
Returns: true

Step 4.
   For all (measure in dataset){
      If "eCons" in {"eCons"}}
Returns: true

Step 5.
   For all (AggregatedFunctions defined in IGOLAP){
      If "SUM" in {"SUM", "AVG", "COUNT", "MIN", "MAX"};}
Returns: true

Step 6. If Step 2 to 5 are true then validation returns true.
```

Applying the F_Roll_up operator with both visualisation and materialisation options, produces the specific SELECT and CONSTRUCT SPARQL queries as illustrated.

```
SELECT DISTINCT  ?Income_label
   ?City_label ?Day_label
   (AVG(?measure) AS ?eCons)
 WHERE
 { ?o rdf:type qb:Observation .
   ?o db1:Household ?topoM .
   ?o db1:City ?City .
   ?o db1:Day ?Day .
   ?o db1:eCons ?measure .
   ?topoM igolap:topoDConnectedTo
   ?Income .
   ?City rdfs:label ?City_label .
   ?Day rdfs:label ?Day_label .
   ?Income rdfs:label
   ?Income_label
}
GROUP BY ?Income_label
   ?City_label ?Day_label
```

```
CONSTRUCT
{?id rdf:type qb:Observation .
 ?id qb:dataSet ds:dataSet_Income_eCons .
 ?id db1:Income ?Income .
 ?id db1:City ?City .
 ?id db1:Day ?Day .
 ?id db1:eCons ?eCons .}
  WHERE { { SELECT DISTINCT  ?Income ?id ?City
   ?Day (AVG(?measure) AS ?eCons)
    WHERE{ ?o rdf:type qb:Observation .
      ?o db1:Household ?topoM .
      ?o db1:City ?City .
      ?o db1:Day ?Day .
      ?o db1:eCons ?measure .
      ?topoM igolap:topoDConnectedTo ?Income
   BIND(iri(concat("http://www.energydb.eu/igola
   p/Data/DataSets#", "_",strafter(str(?Income),
   "#"), "_", strafter(str(?City), "#"), "_",
   strafter(str(?Day), "#"))) AS ?id)
}
GROUP BY ?Income ?id ?City ?Day
}}
```

The outcome of these queries, and the core of the F_Roll_up operator generating them, is presented as follows: the first instance shows the visualisation outcome and the second instance shows a fully new generated dataset.

materialize = false;

```
-------------------------------------------------------------------------------------------
| Income_label                | City_label      | Day_label                    | eCons |
===========================================================================================
| "Income range between
60000 and 70000"@en           | "Birmingham"@en | "10th of March 2011@en"      | 4.98  |
| "Income range between
60000 and 70000"@en           | "Birmingham"@en | "11th of March 2011@en"      | 13.66 |
[..]

| "Income range between
40000 and 50000"@en           | "Birmingham"@en | "10th of February 2011@en" | 19.06 |
| "Income range between
40000 and 50000"@en           | "Birmingham"@en | "11th of April 2011@en"      | 3.22  |
[..]

| "Income range between
10000 and 20000"@en           | "Birmingham"@en | "10th of February 2011@en" | 20.88 |
| "Income range between
10000 and 20000"@en           | "Birmingham"@en | "11th of February 2011@en" | 16.19 |
-------------------------------------------------------------------------------------------
```

materialize = true;

```
ds:DailyHhECons a qb:DataStructureDefinition;
qb:Component [igolap:TopoDimension db1:Income];
qb:Component [qb4o:level db1:City];
qb:Component [qb4o:level db1:Day];
qb:Component [[igolap:TopoDimension db1:Household];
qb:Component [qb4o:measure db1:eCons;
qb4o:hasAggregateFunction qb4o:AVG].

ds:dataSet Income eCons AVG a qb:DataSet;
 qb:structure ds:DailyHhECons.

ds:_ukIrang5_birm_D15M04Y2011
        a           qb:Observation ;
        qb:dataSet  ds:dataSet_Income_eCons ;
```

```
        db1:City    l:birm ;
        db1:Day     t:D15M04Y2011 ;
        db1:Income  topo:ukIrang5 ;
        db1:eCons   2.68 .

ds:_ukIrang5_birm_D03M02Y2011
        a           qb:Observation ;
        qb:dataSet  ds:dataSet_Income_eCons ;
        db1:City    l:birm ;
        db1:Day     t:D03M02Y2011 ;
        db1:Income  topo:ukIrang5 ;
        db1:eCons   14.80 .

ds:_ukIrang7_birm_D05M03Y2011
        a           qb:Observation ;
        qb:dataSet  ds:dataSet Income eCons ;
        db1:City    l:birm ;
        db1:Day     t:D05M03Y2011 ;
        db1:Income  topo:ukIrang7 ;
        db1:eCons   13.36 .

ds: ukIrang2 birm D26M02Y2011
        a           qb:Observation ;
        qb:dataSet  ds:dataSet_Income_eCons ;
        db1:City    l:birm ;
        db1:Day     t:D26M02Y2011 ;
        db1:Income  topo:ukIrang2 ;
        db1:eCons   9.46 .
[..]
```

Query 3: Number of days in a month that have a recorded energy consumption

In this query, the aggregation needs to make use of the COUNT aggregation function that will be applied on the aggregated level Month over its ***childLevels*** Day. Furthermore the topological dimension "Household" requested in the query needs only to be presented in the provided dataset of observations.

The steps are as described in the previous queries, as such, the above stages are only referenced here.

Passed parameters to the F_Roll_up operator:

```
aggLevelDim: "Month", "Time"
measure: "eCons"
measureConstraint: "COUNT"
```

Validation phase with the given parameters:

```
Step 1.retrieve dimensionType of dimension "Time".
   Returns: InfoDimension

Step2.
   If "InfoDimension" equals "InfoDimension" then {
         Retrieve childLevels of required Level;
   }
   Returns: childLevels ={"Day"}
   If "InfoDimension" equals "TopoDimension";
Returns: true

Step 3.
   For all (dimension or levels in dataset) {
      If "Day" in {Day, City, Household}}
```

```
Returns: true

Step 4.
   For all (measure in dataset){
      If "eCons" in {"eCons"}}
Returns: true

Step 5.
   For all (AggregatedFunctions defined in IGOLAP){
      If "SUM" in {"SUM", "AVG", "COUNT", "MIN", "MAX"};}
Returns: true

Step 6. If Step 2 to 5 are true then validation returns true.
```

Generated queries by F_Roll_up operator:

```
SELECT DISTINCT  ?Month_label
   ?City_label ?Household_label
   (COUNT(?measure) AS ?eCons)
 WHERE
 { ?o rdf:type qb:Observation .
   ?o db1:Day ?child .
   ?o db1:City ?City .
   ?o db1:Household ?Household .
   ?o db1:eCons ?measure .
   ?child qb4o:parentLevel ?Month
   .
   ?City rdfs:label ?City_label .
   ?Household rdfs:label
   ?Household_label .
   ?Month rdfs:label ?Month_label
 }
 GROUP BY ?Month_label
   ?City_label ?Household_label
```

```
CONSTRUCT { ?id rdf:type qb:Observation .
   ?id qb:dataSet ds:dataSet_Month_eCons_COUNT .
   ?id db1:Month ?Month .
   ?id db1:City ?City .
   ?id db1:Household ?Household .
   ?id db1:eCons ?eCons .}
    WHERE { { SELECT DISTINCT  ?Month ?id ?City
   ?Household (COUNT(?measure) AS ?eCons)
      WHERE {
      ?o rdf:type qb:Observation .
      ?o db1:Day ?child .
      ?o db1:City ?City .
      ?o db1:Household ?Household .
      ?o db1:eCons ?measure .
      ?child qb4o:parentLevel ?Month

      BIND(iri(concat("http://www.energydb.eu/ig
   olap/Data/DataSets#",
   strafter(str(?Household), "#"), "_",
   strafter(str(?Month), "#"))) AS ?id)
 }
 GROUP BY ?Month ?id ?City ?Household
 }}
```

In this case as well, the "pretty" (well-formatted) version for visualisation of the output is first presented, followed by the materialised output of the same query.

```
materialize = false;
-------------------------------------------------------------------
| Month label         | City label       | Household label | eCons |
===================================================================
| "March 2011@en"     | "Birmingham"@en | "106"@en         | 6     |
| "March 2011@en"     | "Birmingham"@en | "2"@en           | 17    |
| "February 2011@en"  | "Birmingham"@en | "106"@en         | 22    |
| "February 2011@en"  | "Birmingham"@en | "119"@en         | 28    |
| "February 2011@en"  | "Birmingham"@en | "154"@en         | 18    |
| "February 2011@en"  | "Birmingham"@en | "2"@en           | 15    |
| "April 2011@en"     | "Birmingham"@en | "119"@en         | 15    |
-------------------------------------------------------------------
```

```
materialize = true;
ds:DailyHhECons a qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [qb4o:level db1:Day];
qb:Component [[igolap:TopoDimension db1:Household];
qb:Component [qb4o:measure db1:eCons;
```

```
qb4o:hasAggregateFunction qb4o:COUNT].

ds:dataSet_Month_eCons_COUNT a qb:DataSet;
 qb:structure ds:DailyHhECons.

ds:hh2_M02Y2011  a     qb:Observation ;
        qb:dataSet     ds:dataSet_Month_eCons_COUNT ;
        db1:City       l:birm ;
        db1:Household  topo:hh2 ;
        db1:Month      t:M02Y2011 ;
        db1:eCons      15 .

ds:hh154_M02Y2011  a   qb:Observation ;
        qb:dataSet     ds:dataSet_Month_eCons_COUNT ;
        db1:City       l:birm ;
        db1:Household  topo:hh154 ;
        db1:Month      t:M02Y2011 ;
        db1:eCons      18 .

ds:hh106 M03Y2011  a   qb:Observation ;
        qb:dataSet     ds:dataSet Month eCons COUNT ;
        db1:City       l:birm ;
        db1:Household  topo:hh106 ;
        db1:Month      t:M03Y2011 ;
        db1:eCons      6 .

ds:hh2 M03Y2011  a     qb:Observation ;
        qb:dataSet     ds:dataSet_Month_eCons_COUNT ;
        db1:City       l:birm ;
        db1:Household  topo:hh2 ;
        db1:Month      t:M03Y2011 ;
        db1:eCons      17 .
```

6.3.2 F_DRILL Operator

6.3.2.1 Implementation

In order to achieve the expected functionality of a drill-down operator, the F_Drill operator that is introduced in this work needs to navigate twice through the datasets. In the first instance, it needs to retrieve the data structure definition of the current set and then find the matching observation describing the deeper level of one dimension, while still connected to the dimensions from the initial set.

For example, in the code from Extract 7 it is exemplified the data structure definition from input data and the targeted data structure definition after the drill-down operation.

Extract 7:

Input dataset data structure	Targeted dataset data structure
ds:DailyHhECons a qb:DataStructureDefinition; qb:Component [qb4o:level db1:Month]; qb:Component [qb4o:level db1:City]; qb:Component [igolap:TopoDimension db1:Household]; qb:Component [qb4o:measure db1:eCons; qb4o:hasAggregateFunction qb4o:SUM].	ds:DailyHhECons a qb:DataStructureDefinition; qb:Component [qb4o:level db1:Day]; qb:Component [qb4o:level db1:City]; qb:Component [igolap:TopoDimension db1:Household]; qb:Component [qb4o:measure db1:eCons].

The only difference between the above structures is the level of details on one of the informational dimensions, in the above case.

Since the F_Drill operator is implemented to retrieve the dataset structure and members from an input dataset and to retrieve the members representing at a deeper granularity with the same data structure, it needs to have access to two different set of data. Firstly to the input dataset and secondly to the entire datasets from the database in order to identify the relevant observations.

In order to deliver this, the constructor has three data input parameters as:

- location of the database schema
- initial aggregation location
- location of all other datasets in the targeted database

In the case of the ***validate()*** method, this is implemented in the same way as in the F_Roll_up operator. Consequently this is not illustrated in this section, but the implementation source code is made available in the Appendix B of this work, together with the entire F_Drill operator and all other Federated operators implementation, available to download from the online repository provided.

In the case of the ***calculate()*** method, delivering the SELECT and CONSTRUCT computation, the handling of the topological and informational dimensions require only one difference in the SELECT construction. For this reason, the construction of the SELECT, both for visualising or input for the CONSTRUCT, is illustrated for the

informational dimension handling. Nevertheless, the change required for topological dimension handling is highlighted and provided as well.

```
public static void calculate(String db_name, Model schemasModel, Model dataModel, String
   measure, boolean materialize) {
if (dimensionType.equalsIgnoreCase(DB1_Vocabulary.igolap + "InfoDimension")) {
try {
   List<String> iLevels = new ArrayList<String>();
   List<String> tDims = new ArrayList<String>();
   String q1 = "";
   String q3a = "";
   String d_q3a ="";
   String q4 = "";
   String q1Visual = "";
   String q4Visual = "";
   for (int i = 0; i < infoLevels.size(); i++) {
      iLevels.add(infoLevels.get(i).replaceAll(DB1_Vocabulary.db1, ""));
      if (!(infoLevels.get(i).equalsIgnoreCase(levelsParent.get(0)))) {
         q1= q1 + " ?" + iLevels.get(i);
         q3a= q3a + db_name + ":" + iLevels.get(i) + " ?"+ iLevels.get(i) + ";";
         d_q3a =d_q3a+db_name+":"+iLevels.get(i)+" ?"+ iLevels.get(i) + "_cons ;";
         q1Visual=q1Visual + " ?" + iLevels.get(i) + "_label";
         q4Visual=q4Visual+"?"+iLevels.get(i)+" rdfs:label ?"+iLevels.get(i)+"_label .";
      }
   }
   for (int i = 0; i < topos.size(); i++) {
      tDims.add(topos.get(i).replaceAll(DB1_Vocabulary.db1, ""));
      q1 = q1 + " ?" + tDims.get(i);
      q3a = q3a + db_name + ":" + tDims.get(i) + " ?"       + tDims.get(i) + ";";
      d_q3a = d_q3a +db_name+":"+tDims.get(i)+" ?"+tDims.get(i)+ "_cons ;";
      q1Visual = q1Visual +" ?" + tDims.get(i)+ "_label";
      q4Visual = q4Visual+"?" +tDims.get(i)+" rdfs:label ?"+ tDims.get(i)+"_label .";
   }
   String q1a = " ?" + measure + " ";
   String q3b = d_q3a+db_name +":"+measure+" ?measure_cons ."+"?o2 a qb:Observation; "+
   db_name+":"+rLevel+" ?"+rLevel+"; "+q3a+ db_name+ ":" +measure+" ?"+measure+" .";
   String q0 = "SELECT DISTINCT ";
   String q0Constr = q0 + "?" + rLevel + " ?id";
   String q0Visual = q0 + "?" + rLevel + "_label" ;
   String q2 = " WHERE {" + "?o a qb:Observation; " ;
   String q3 = "<" + levelsParent.get(0) + "> ?parent; " ;
   q4 = "?parent igolap:childLevel ?" + rLevel + ". " ;
   q4Visual = q4Visual  + "?" + rLevel + " rdfs:label?" + rLevel + "_label .";
   String q5 = "} GROUP BY ?"+measure+" ";
   String q5Constr = q5 + "?" + rLevel + " ?id" + q1;
   String q5Visual = q5 + "?" + rLevel + "_label" + q1Visual;
   String q4Constr = "BIND (iri(concat(\"" + DB1_Vocabulary.ds+ "\",STRAFTER(str(?" +
   tDims.get(0) + "), \"#\"),\"_\",STRAFTER(str(?" + rLevel + "),\"#\"))) AS ?id)";
   String SelectConstruct = q0Constr+q1+ q1a+ q2+ q3+ q3b+ q4+ q4Constr+ q5Constr;
   String SelectVisualize = q0Visual+q1Visual+q1a+q2+ q3+q3b+ q4+ q4Visual+ q5Visual;
```

In the case of topological dimension, the highlighted section is implemented as provided below:

```
q4 = "?topoM igolap:topoDConnectedTo ? " + rLevel + " .";
```

In regards to the CONSTRUCT, there is no change in the construction of the SPARQL operator. With all these, there is a change in regards of data structure definition accompanying the newly obtained dataset.

The implementation of the CONSTRUCT is introduced below:

```
if (materialize) {
   // CONSTRUCT
   String constructString = prefix + NL + "CONSTRUCT { "
      + "?id a qb:Observation; qb:dataSet ds:dataSet_"
      + rLevel + "_" + measure +  " ; " + db_name + ":" + rLevel + " ?" + rLevel
      + "; " + q3a + " " + db_name + ":" + measure + " ?"
      + measure + "}" + "WHERE{" + " {"      + SelectConstruct + "}" + "}";

   Query constructQuery = QueryFactory.create(constructString);
   QueryExecution constr = QueryExecutionFactory.create(constructQuery, m2);
   OutputStream output = new FileOutputStream(dataWrite_path);
   obsModel.write(output, "N3", null);
```

With regards to the data structure definition, again there is only one change required for handling the two types of dimensions. As such, only one implementation is introduced. Time the topological dimension is used for illustration, with highlight on the required change. The implementation of the change is also provided.

```
String s = "ds:DailyHhECons a qb:DataStructureDefinition;"+ NL
      + "qb:Component [igolap:TopoDimension " + db_name + ":" + rLevel + "];";
   for (int j = 0; j < iLevels.size(); j++) {
      s = s + "qb:Component [qb4o:level " + db_name + ":"   + iLevels.get(j) + "];" ;
      }
   for (int j = 0; j < tDims.size(); j++) {
      if (!(tDims.get(j).equalsIgnoreCase(rDim))) {
         s = s +"qb:Component [igolap:TopoDimension "+db_name +":"+tDims.get(j)+"];"+NL;
         }
      }
   s = s + "qb:Component [qb4o:measure " + db_name + ":"+ measure +  "]." + NL;
   String datasetSchema = s + "ds:123ataset_" + rLevel + "_"+ measure + " a qb:DataSet;"
      + NL+ " qb:structure ds:DailyHhECons.";
   output.write(datasetSchema.getBytes(Charset.forName("UTF-8")));
```

In order to build the data structure definition of the informational dimension, the change in the highlighted portion it is the following:

```
String s = "ds:DailyHhECons a qb:DataStructureDefinition;"+ NL
    + "qb:Component [qb4o:level " + db_name + ":" + rLevel + "];";
```

Each of the SELECT and CONSTRUCT are exemplified in the following section.

6.3.2.2 Exemplification

In order to exemplify the F_Drill operators a set of queries was selected. In the previous section, it was mentioned that the differences are limited and only on the level of handling topological and informational dimensions. As the drill operator operates on only one dimension at a time, it is sufficient and relevant to have two queries defined, one for drill operation on a topological dimension and one for drill operation on an informational dimension.

These two queries are exemplified for both a visualisation (SELECT only) request as well as for a materialisation (CONSTRUCT) request.

Query 4: Retrieve the days that had an energy consumption input registered, and their values, used to aggregate the monthly energy consumption.

In this query the input dataset, or the start point dataset is actually the output dataset from ***Query 1***.

The passed parameters to the operator's call are:

```
deAggLevelDim: "Day", "Time"
```

Given these parameters and the required datasets paths, the F_Drill operator computes the following SELECT and CONSTRUCT SPARQL queries:

```
SELECT DISTINCT ?Day_label
   ?City_label ?Household_label
   ?eCons
 WHERE {
?o rdf:type qb:Observation .
?o db1:Month ?parent .
?o db1:City ?City_cons .
?o db1:Household ?Household_cons.
?o db1:eCons ?measure_cons .
?o2 rdf:type qb:Observation .
?o2 db1:Day ?Day .
?o2 db1:City ?City .
?o2 db1:Household ?Household .
?o2 db1:eCons ?eCons .
?parent igolap:childLevel ?Day .
?City rdfs:label ?City_label .
?Household rdfs:label
   ?Household_label .
?Day rdfs:label ?Day_label
 } GROUP BY ?eCons ?Day_label
   ?City_label ?Household_label
```

```
CONSTRUCT { ?id rdf:type qb:Observation .
?id qb:dataSet ds:dataSet_Day_eCons .
?id db1:Day ?Day .
?id db1:City ?City .
?id db1:Household ?Household .
?id db1:eCons ?eCons .}
 WHERE{ {SELECT DISTINCT  ?Day ?id ?City
   ?Household ?eCons
 WHERE
 { ?o rdf:type qb:Observation .
   ?o db1:Month ?parent .
   ?o db1:City ?City_cons .
   ?o db1:Household ?Household_cons .
   ?o db1:eCons ?measure_cons .
   ?o2 rdf:type qb:Observation .
   ?o2 db1:Day ?Day .
   ?o2 db1:City ?City .
   ?o2 db1:Household ?Household .
   ?o2 db1:eCons ?eCons .
   ?parent igolap:childLevel ?Day
   BIND(iri(concat("http://www.energydb.eu/igola
   p/Data/DataSets#", strafter(str(?Household),
   "#"), "_", strafter(str(?Day), "#"))) AS ?id)
 }
 GROUP BY ?eCons ?Day ?id ?City ?Household }}
```

Based on these SPARQL queries we have the output as for the SELECT only marked as materialize=false and for CONSTRUCT true. A sample of this outputs is provided below:

```
materialize = false;
-------------------------------------------------------------------------
| Day label                 | City label      | Household label | eCons |
=========================================================================
| "15th of February 2011@en" | "Birmingham"@en | "119"@en        | 14.94 |
| "17th of March 2011@en"    | "Birmingham"@en | "2"@en          | 12.89 |
| "21st of February 2011@en" | "Birmingham"@en | "106"@en        | 6.19  |
| "27th of February 2011@en" | "Birmingham"@en | "154"@en        | 6.64  |
| "6th of March 2011@en"     | "Birmingham"@en | "2"@en          | 6.02  |
| "2nd of February 2011@en"  | "Birmingham"@en | "2"@en          | 16.67 |
```

```
| "16th of February 2011@en" | "Birmingham"@en | "119"@en        | 12.59 |
[..]
```

```
materialize = true;
ds:DailyHhECons a qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension db1:Household];
qb:Component [qb4o:measure db1:eCons].

ds:dataSet Day eCons a qb:DataSet;
 qb:structure ds:DailyHhECons.

ds:hh119_D05M02Y2011  a  qb:Observation ;
        qb:dataSet     ds:dataSet_Day_eCons ;
        db1:City       l:birm ;
        db1:Day        t:D05M02Y2011 ;
        db1:Household  topo:hh119 ;
        db1:eCons      12.55 .

ds:hh119 D20M02Y2011  a  qb:Observation ;
        qb:dataSet     ds:dataSet Day eCons ;
        db1:City       l:birm ;
        db1:Day        t:D20M02Y2011 ;
        db1:Household  topo:hh119 ;
        db1:eCons      7.91 .

ds:hh106 D16M02Y2011  a  qb:Observation ;
        qb:dataSet     ds:dataSet Day eCons ;
        db1:City       l:birm ;
        db1:Day        t:D16M02Y2011 ;
        db1:Household  topo:hh106 ;
        db1:eCons      5.03 .
[..]
```

Query 5: From the daily average energy consumption per Income, retrieve the households and their daily energy consumption for the recorded Income ranges.

For this query the input dataset, or the start point dataset, is the output dataset from ***Query 2***. This query requires a navigation through topological dimensions. From an ***Income*** dimension view to a ***Household*** dimension view, while keeping the other dimensions at the same level.

For retrieving topological dimension's member, it is needed as in the case of F_Roll_up, to be given the two topological dimensions, describing **to** which **from** which topological dimension I want to navigate.

```
deAggTopoDim: "Household", "Income"
```

Using these parameters and the required datasets paths, the F_Drill operator computes the below SELECT and CONSTRUCT SPARQL queries:

```
SELECT DISTINCT  ?Household_label
   ?Day_label ?City_label ?eCons
 WHERE { ?o rdf:type
   qb:Observation .
   ?o db1:Income ?topoM .
   ?o db1:Day ?Day_cons .
   ?o db1:City ?City_cons .
   ?o db1:eCons ?measure_cons .
   ?o2 rdf:type qb:Observation .
   ?o2 db1:Household ?Household .
   ?o2 db1:Day ?Day .
   ?o2 db1:City ?City .
   ?o2 db1:eCons ?eCons .
   ?topoM igolap:topoDConnectedTo
?Household .
?City rdfs:label ?City_label .
?Household rdfs:label
   ?Household_label .
?Day rdfs:label ?Day_label
 } GROUP BY ?eCons ?Day_label
   ?City_label ?Household_label
```

```
CONSTRUCT { ?id rdf:type qb:Observation .
?id qb:dataSet ds:dataSet_Household_eCons .
?id db1:Household ?Household .
?id db1:Day ?Day .
?id db1:City ?City .
?id db1:eCons ?eCons .}
 WHERE {{ SELECT DISTINCT  ?Household ?id ?Day
   ?City ?eCons
 WHERE { ?o rdf:type qb:Observation .
   ?o db1:Income ?topoM .
   ?o db1:Day ?Day_cons .
   ?o db1:City ?City_cons .
   ?o db1:eCons ?measure_cons .
   ?o2 rdf:type qb:Observation .
   ?o2 db1:Household ?Household .
   ?o2 db1:Day ?Day .
   ?o2 db1:City ?City .
   ?o2 db1:eCons ?eCons .
   ?topoM igolap:topoDConnectedTo
?Household .
   BIND(iri(concat("http://www.energydb.eu/igola
   p/Data/DataSets#", "_",
   strafter(str(?Household), "#"), "_",
   strafter(str(?Day), "#"), "_",
   strafter(str(?City), "#"))) AS ?id)
 } GROUP BY ?eCons ?Household ?id ?Day ?City }}
```

The outcomes of the above computed SPARQL queries will generate in this case the outcome which is sampled below:

materialize = false;

```
-------------------------------------------------------------------------
| Household_label | Day_label                  | City_label      | eCons |
=========================================================================
| "119"@en        | "15th of February 2011@en" | "Birmingham"@en | 14.94 |
| "2"@en          | "17th of March 2011@en"    | "Birmingham"@en | 12.89 |
| "106"@en        | "21st of February 2011@en" | "Birmingham"@en | 6.19  |
| "154"@en        | "27th of February 2011@en" | "Birmingham"@en | 6.64  |
| "2"@en          | "6th of March 2011@en"     | "Birmingham"@en | 6.02  |
[..]
```

materialize = true;

```
ds:DailyHhECons a qb:DataStructureDefinition;
qb:Component [igolap:TopoDimension db1:Household];
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [qb4o:measure db1:eCons].

ds:dataSet Household eCons a qb:DataSet;
 qb:structure ds:DailyHhECons.

ds:_hh106_D09M02Y2011_birm
        a              qb:Observation ;
        qb:dataSet     ds:dataSet Household eCons ;
        db1:City       l:birm ;
        db1:Day        t:D09M02Y2011 ;
        db1:Household  topo:hh106 ;
        db1:eCons      6.73 .

ds: hh2 D28M02Y2011 birm
        a              qb:Observation ;
        qb:dataSet     ds:dataSet Household eCons ;
        db1:City       l:birm ;
        db1:Day        t:D28M02Y2011 ;
        db1:Household  topo:hh2 ;
        db1:eCons      13.46 .
```

```
ds:_hh106_D20M02Y2011_birm
        a               qb:Observation ;
        qb:dataSet      ds:dataSet Household eCons ;
        db1:City        l:birm ;
        db1:Day         t:D20M02Y2011 ;
        db1:Household   topo:hh106 ;
        db1:eCons       3.92 .
[..]
```

6.3.3 F_SLICE Operator

6.3.3.1 Implementation

In the case of the Slice operators, since it operates on the members' level from a given cube or dataset, it is not relevant for the SELECT operator if the members required for the Slice operation come from a topological or from an informational dimension.

Since TopoDimensions do not have levels, the members in this case belong to the dimension itself. This is allowed due to the characteristics introduced through the defined IGOLAP Vocabulary.

In the case of F_Slice operator, the ***validate()*** method has a different focus. This is to identify that the required members and measures are well defined and available in the data so that the operation can be performed. Since a member can belong to a topological dimension directly or to a level in an informational dimension, the one validation of a member's declaration in the method is as follows.

```
public static boolean validate(String db_name, HashMap<String, String> aggMemberLevel,
   Model schemasModel, Model dataModel) {
   Iterator<String> it = aggMemberLevel.keySet().iterator();
   if (it.hasNext()) {
      rMember = it.next();
      rLevel = aggMemberLevel.get(rMember);
      try {
         String levelVal =prefix + NL
                     + "SELECT ?lType WHERE {?ds a qb:DataStructureDefinition;"
                     + "?p ?o . ?o ?lType "+db_name+":"+rLevel+ "}";
         Query queryDimension = QueryFactory.create(levelVal);
         QueryExecution qExe = QueryExecutionFactory.create(queryDimension, dataModel);
         ResultSet levelsType = qExe.execSelect();
         if (levelsType.hasNext()){
         QuerySolution solLevel = levelsType.next();
         String c = solLevel.get("lType").toString().replaceAll(DB1_Vocabulary.qb, "");
         c = c.replaceAll(DB1_Vocabulary.igolap, "");
         c= c.replaceAll(DB1_Vocabulary.qb4o, "");
         if (c!=null){
             switch (c) {
                   case "level":  levelType = "InfoDimension"; break;
```

```
                    case "TopoDimension":  levelType = "TopoDimension";break;
                    default:  System.out.println("Invalid Vocab used!");break;
                }
              }
           }else{
              System.out.println("Level doesn't exist!");
              return false;
           }
           String selectTopoString = prefix + NL + "SELECT ?TopoDims "
                 + "WHERE {?ds a qb:DataStructureDefinition;"
                 + "?p ?o . ?o igolap:TopoDimension ?TopoDims" + " }";
           Query selectTopoComp = QueryFactory.create(selectTopoString);
           QueryExecution retrieveTopoComp = QueryExecutionFactory.create(selectTopoComp,
   dataModel);
           ResultSet topoRes = retrieveTopoComp.execSelect();

           while (topoRes.hasNext()) {
              QuerySolution qs = topoRes.next();
              topos.add(qs.get("TopoDims").toString());
           }
           String selectInfoLevelString = prefix + NL + "SELECT ?InfoLvls "
                 + "WHERE {?ds a qb:DataStructureDefinition;"
                 + "?p ?o . ?o qb4o:level ?InfoLvls" + " }";
           Query selectInfoComp = QueryFactory.create(selectInfoLevelString);
           QueryExecution retrieveInfoComp = QueryExecutionFactory.create(
                        selectInfoComp, dataModel);
           ResultSet infoRes = retrieveInfoComp.execSelect();

           while (infoRes.hasNext()) {
              QuerySolution qs = infoRes.next();
              infoLevels.add(qs.get("InfoLvls").toString());
           }
           String selectMeasureString = prefix + NL + "SELECT ?measure "
                 + "WHERE {?ds a qb:DataStructureDefinition;"
                 + "?p ?o . ?o qb4o:measure ?measure }";
           Query selectMeasureComp = QueryFactory.create(selectMeasureString);
           QueryExecution retrieveMeasureComp = QueryExecutionFactory.create(
                 selectMeasureComp, dataModel);
           ResultSet measureRes = retrieveMeasureComp.execSelect();
           while (measureRes.hasNext()) {
              QuerySolution qs = measureRes.next();
              String found_measure = qs.get("measure").toString();
              measures.add(found_measure);
           }
           return true;
           } catch (Exception e) {
           e.printStackTrace();
           System.out.println("Dimensions retrieval exception!");
           return false;
           }
        }else{return false;}
     }
```

The SELECT and CONSTRUCT operators implementation is described below, for the two options: visualisation and materialisation.

```
public static void calculate(String db_name, Model schemasModel, Model dataModel,
   boolean materialize) {
//type of the level/dimension is not relevant here as all members are igolap:Member type
   try {
   Model m2 = ModelFactory.createUnion(schemasModel, dataModel);
   [..]
   for (int i = 0; i < infoLevels.size(); i++) {
      iLevels.add(infoLevels.get(i).replaceAll(Vocabulary.db1, ""));
   if (!(iLevels.get(i).equalsIgnoreCase(rLevel))){
      q1 = q1+" ?"+iLevels.get(i);
      q3a = q3a+db_name+":"+iLevels.get(i)+" ?"+iLevels.get(i)+";";
      q1Visual=q1Visual+" ?"+iLevels.get(i)+"_label";
      q4Visual= q4Visual+"?"+iLevels.get(i)+" rdfs:label ?"+iLevels.get(i)+"_label.";}
      }
   for (int i = 0; i < topos.size(); i++) {
```

```
        tDims.add(topos.get(i).replaceAll(Vocabulary.db1, ""));
        if (!(tDims.get(i).equalsIgnoreCase(rLevel))) {
        q1 = q1+" ?"+tDims.get(i);
        q3a = q3a+db_name+":"+tDims.get(i)+" ?"+tDims.get(i)+";";
        q1Visual=q1Visual+" ?"+tDims.get(i)+"_label";
        q4Visual= q4Visual+"?"+tDims.get(i)+" rdfs:label ?"+tDims.get(i)+"_label . ";}
    }
    String q1a = " ?"+measure;
    String q3b=q3a+db_name+":"+measure+" ?"+measure+" .";

    String q0 = "SELECT DISTINCT ";
    String q0Constr= q0+"?requested_"+rLevel+" ?id";
    String q0Visual =q0+"?requested_"+rLevel+"";
    String q2 = " WHERE {" ?o a qb:Observation; ";
    String q3= "db1:"+rLevel+" "+rMember+ "; ";
    q4Visual =q4Visual+rMember+" rdfs:label ?requested_"+rLevel+" .";
    String q5 ="} GROUP BY ";
    String q5Constr = q5+"?requested_"+rLevel+" ?id"+q1+q1a ;
    String q5Visual = q5+"?requested_"+rLevel+" "+q1Visual+q1a;
    String q4Constr="BIND (iri(concat(\""+Vocabulary.ds+"\",STRAFTER(str(?"+
    tDims.get(0)+"), \"#\"),\"_\",STRAFTER(str("+rMember+"),\"#\"))) AS ?id)";

    String SelectConstruct = q0Constr+q1+q1a+q2+q3+q3b+q4+q4Constr+q5Constr;
    String SelectVisualize = q0Visual+q1Visual+q1a+q2+q3+q3b+q4+q4Visual+q5Visual;
```

The CONSTRUCT operator for the F_SLICE operators has the following implementation:

```
if (materialize) {
    String constructString = prefix + NL + "CONSTRUCT { " + "?id a qb:Observation; "+
    "qb:dataSet ds:dataSet_"+ rLevel+"_" +measure+"_" +rMember.replaceAll("t:", "") +" ;
    "+db_name+":"+rLevel+" "+rMember+"; "+q3a+" "+db_name+":"+measure+" ?"+measure+"}"
        + "WHERE{ {"
        + SelectConstruct
        + "} }";
    Query constructQuery = QueryFactory.create(constructString);
    QueryExecution constr = QueryExecutionFactory.create(constructQuery, m2);
    Model obsModel = constr.execConstruct();
    System.out.println("The construct obtained:");
    obsModel.write(System.out);
    OutputStream output = new FileOutputStream(dataWrite_path);
    obsModel.write(output, "N3", null);
    String s= "ds:DailyHhECons a qb:DataStructureDefinition;"+NL;
        for (int j = 0; j < iLevels.size(); j++) {
            s= s+"qb:Component [qb4o:level "+db_name+":"+iLevels.get(j)+"];"+NL;}
        for (int j = 0; j < tDims.size(); j++) {
            s= s+"qb:Component [igolap:TopoDimension "+db_name+":"+tDims.get(j)+"];"+NL;}
    s=s+"qb:Component [qb4o:measure "+db_name+":"+measure+"]."+NL;
    String datasetSchema=s+"ds:dataSet_"+rLevel+"_"+measure+"_"+rMember.replaceAll("t:",
    "")+" a qb:DataSet;"+NL+" qb:structure ds:DailyHhECons." ;
    output.write(datasetSchema.getBytes(Charset.forName("UTF-8")));
```

6.3.3.2 Exemplification

In order to exemplify the SLICE operators, two queries requiring slice aggregations on both topological and informational dimensions are introduced in this section. The F_Slice operator is used to deliver these aggregations on the given dataset; the database schema; and, the members definition described previously. Taking the parameters required for these queries, the output from the ***validate()*** and the ***calculate()*** methods will also be exemplified.

The dataset used in these queries is the dataset obtained from applying F_Roll_up as requested by ***Query 1***. The structure of the dataset used is presented below and the entire extract used for this exemplification can be found in Appendix B2.2.

```
ds:hh119 M02Y2011  a   qb:Observation ;
        qb:dataSet     ds:dataSet Month eCons SUM ;
        db1:City       l:birm ;
        db1:Household  topo:hh119 ;
        db1:Month      t:M02Y2011 ;
        db1:eCons      395.63 .
```

Since in the F_Slice case, the validation of a member is easy to follow without a detailed examination of the ***validate()*** method, this will be omitted for the F_Slice queries in this section.

The F_Slice operator is exemplified in the following paragraphs through two queries.

Query 6: Extract the February month from the monthly energy consumption of the given dataset (cube).

In this query, all the months except February need to be removed from the dataset, or in other words, a new dataset containing only the level Month – member February needs to be created. In the case of the Slice operator, as presented in the definition above, the structure of the dataset is maintained. As a consequence, this member is presented in conjunction with all other members from other dimensions, to which it is related.

Passed parameters to the F_Slice_up operator:

```
aggMemberLevel: "t:M02Y2011", “Time”
```

Generate SPARQL queries by F_Slice operator for Query 6:

```
SELECT DISTINCT  ?requested_Month
   ?City_label ?Household_label
   ?eCons
  WHERE{
   ?o rdf:type qb:Observation .
   ?o db1:Month t:M02Y2011 .
   ?o db1:City ?City .
   ?o db1:Household ?Household .
   ?o db1:eCons ?eCons .
   ?City rdfs:label ?City_label .
   ?Household rdfs:label
   ?Household_label .
   t:M02Y2011 rdfs:label
   ?requested_Month
 }
 GROUP BY ?requested_Month
   ?City_label ?Household_label
   ?eCons
```

```
CONSTRUCT
 { ?id rdf:type qb:Observation .
   ?id qb:dataSet
   ds:dataSet_Month_eCons_M02Y2011 .
   ?id db1:Month t:M02Y2011 .
   ?id db1:City ?City .
   ?id db1:Household ?Household .
   ?id db1:eCons ?eCons .}
 WHERE { { SELECT DISTINCT  ?id ?City
   ?Household ?eCons
 WHERE {
   ?o rdf:type qb:Observation .
   ?o db1:Month t:M02Y2011 .
   ?o db1:City ?City .
   ?o db1:Household ?Household .
   ?o db1:eCons ?eCons

   BIND(iri(concat("http://www.energydb.eu/ig
   olap/Data/DataSets#", "_",
   strafter(str(t:M02Y2011), "#"), "_",
   strafter(str(?City), "#"), "_",
   strafter(str(?Household), "#"))) AS ?id)
 }
 GROUP BY ?id ?City ?Household ?eCons
 } }
```

After running the above SPARQL queries, the outcome is presented below as with materialized option for the CONSTRUCT and with the materialized being false for the SELECT:

materialize = false;

requested_Month	City_label	Household_label	eCons
"February 2011@en"	"Birmingham"@en	"2"@en	231.07
"February 2011@en"	"Birmingham"@en	"119"@en	395.63
"February 2011@en"	"Birmingham"@en	"106"@en	140.02
"February 2011@en"	"Birmingham"@en	"154"@en	185.50

materialize = true;

```
ds:DailyHhECons a qb:DataStructureDefinition;
qb:Component [qb4o:level db1:City];
qb:Component [qb4o:level db1:Month];
qb:Component [igolap:TopoDimension db1:Household];
qb:Component [qb4o:measure db1:eCons].
ds:dataSet Month eCons M02Y2011 a qb:DataSet;
 qb:structure ds:DailyHhECons.

ds:_M02Y2011_birm_hh106
        a               qb:Observation ;
        qb:dataSet      ds:dataSet_Month_eCons_M02Y2011 ;
        db1:City        l:birm ;
        db1:Household   topo:hh106 ;
        db1:Month       t:M02Y2011 ;
        db1:eCons       140.02 .

ds:_M02Y2011_birm_hh154
        a               qb:Observation ;
        qb:dataSet      ds:dataSet_Month_eCons_M02Y2011 ;
        db1:City        l:birm ;
```

```
        db1:Household  topo:hh154 ;
        db1:Month      t:M02Y2011 ;
        db1:eCons      185.50 .
```

It can be observed that the schema in the materialized output of Query 6 and the one of the initial dataset (output of Query1) remained the same.

Query 7: Extract a specific household' monthly consumption from a give dataset.

In this case, the member belongs to the topological dimension Household, and the level of detail in the input dataset is monthly consumption. The input in this case, can also be the same as in the query above, the one generated by the F_Roll_up operator from Query1.

Below is presented the member from the Household topological dimension, passed to the operator as a parameter:

```
aggMemberLevel: "topo:hh119", "Household"
```

Generate SPARQL queries by F_Slice operator for Query 7:

```
SELECT DISTINCT
   ?requested_Household ?City_label
   ?Month_label ?eCons
WHERE
{ ?o rdf:type qb:Observation .
  ?o db1:Household topo:hh119 .
  ?o db1:City ?City .
  ?o db1:Month ?Month .
  ?o db1:eCons ?eCons .
  ?City rdfs:label ?City_label .
  ?Month rdfs:label ?Month_label .
  topo:hh119 rdfs:label
   ?requested_Household
   }
 GROUP BY ?requested_Household
   ?City_label ?Month_label ?eCons
```

```
CONSTRUCT
 { ?id rdf:type qb:Observation .
   ?id qb:dataSet
   ds:dataSet_Month_eCons_M02Y2011 .
   ?id db1:Month t:M02Y2011 .
   ?id db1:City ?City .
   ?id db1:Household ?Household .
   ?id db1:eCons ?eCons .}
 WHERE { { SELECT DISTINCT  ?id ?City
   ?Household ?eCons
 WHERE {
   ?o rdf:type qb:Observation .
   ?o db1:Month t:M02Y2011 .
   ?o db1:City ?City .
   ?o db1:Household ?Household .
   ?o db1:eCons ?eCons

   BIND(iri(concat("http://www.energydb.eu/ig
   olap/Data/DataSets#", "_",
   strafter(str(t:M02Y2011), "#"), "_",
   strafter(str(?City), "#"), "_",
   strafter(str(?Household), "#"))) AS ?id)
 }
 GROUP BY ?id ?City ?Household ?eCons
 } }
```

After running the above SPARQL queries, the outcome is presented below as with materialized option for the CONSTRUCT and with the materialized being false for the SELECT:

```
materialize = false;
-------------------------------------------------------------------------
| requested_Household | City_label      | Month_label        | eCons  |
=========================================================================
| "119"@en            | "Birmingham"@en | "February 2011@en" | 395.63 |
| "119"@en            | "Birmingham"@en | "April 2011@en"    | 31.30  |
-------------------------------------------------------------------------
```

```
materialize = true;
ds:_hh119_birm_M04Y2011
        a              qb:Observation ;
        qb:dataSet     ds:dataSet Household eCons hh119 ;
        db1:City       l:birm ;
        db1:Household  topo:hh119 ;
        db1:Month      t:M04Y2011 ;
        db1:eCons      31.30 .

ds:_hh119_birm_M02Y2011
        a              qb:Observation ;
        qb:dataSet     ds:dataSet_Household_eCons_hh119 ;
        db1:City       l:birm ;
        db1:Household  topo:hh119 ;
        db1:Month      t:M02Y2011 ;
        db1:eCons      395.63 .
```

6.3.4 F_DICE Operator

6.3.4.1 Implementation

The F_Dice operator, guided by the definition above, is implemented as a slice operation across multiple dimensions. The main difference between F_Slice and F_Dice is that in the later the only dimensions kept are the ones relating to a member parameter passed in the operator's call. The operator receives as input a HashMap object of each ***member-level*** or ***member-dimension*** combination that needs to be aggregated.

In conclusion, the parameter takes a set of members as input, and retrieves from all the dimensions to which a member belongs from the given dataset.

The ***validate()*** method in this case is the same as for the F_Slice operator, with the main difference that a set of members, not just one, are each validated in a single call.

```
public static boolean validate(String db_name, HashMap<String, List<String>>
  aggLevelMembers, Model schemasModel, Model dataModel) {
  try {
     String selectTopoString = prefix + NL + "SELECT ?TopoDims "
           + "WHERE {?ds a qb:DataStructureDefinition;"
           + "?p ?o . ?o igolap:TopoDimension ?TopoDims" + " }";
     Query selectTopoComp = QueryFactory.create(selectTopoString);
     QueryExecution retrieveTopoComp = QueryExecutionFactory.create(selectTopoComp,
  dataModel);
     ResultSet topoRes = retrieveTopoComp.execSelect();
     while (topoRes.hasNext()) {
```

```
            QuerySolution qs = topoRes.next();
            topos.add(qs.get("TopoDims").toString());
        }
        String selectInfoLevelString = prefix + NL + "SELECT ?InfoLvls "
              + "WHERE {?ds a qb:DataStructureDefinition;"
              + "?p ?o . ?o qb4o:level ?InfoLvls" + " }";
        Query selectInfoComp = QueryFactory.create(selectInfoLevelString);
        QueryExecution retrieveInfoComp = QueryExecutionFactory.create(
                  selectInfoComp, dataModel);
        ResultSet infoRes = retrieveInfoComp.execSelect();

        while (infoRes.hasNext()) {
            QuerySolution qs = infoRes.next();
            infoLevels.add(qs.get("InfoLvls").toString());
        }
        String selectMeasureString = prefix + NL + "SELECT ?measure "
              + "WHERE {?ds a qb:DataStructureDefinition;"
              + "?p ?o . ?o qb4o:measure ?measure }";
        Query selectMeasureComp = QueryFactory.create(selectMeasureString);
        QueryExecution retrieveMeasureComp = QueryExecutionFactory.create(
              selectMeasureComp, dataModel);
        ResultSet measureRes = retrieveMeasureComp.execSelect();
        while (measureRes.hasNext()) {
            QuerySolution qs = measureRes.next();
            String found_measure = qs.get("measure").toString();
            measures.add(found_measure);
        }
        String rLevel;
        buildComponent = "ds:DailyHhECons a qb:DataStructureDefinition;";
        Iterator<String> it = aggLevelMembers.keySet().iterator();
        while (it.hasNext()) {
            boolean levelExists = false;
            rLevel = it.next();
            for (int i = 0; i < infoLevels.size(); i++) {
            if (rLevel.equalsIgnoreCase(infoLevels.get(i).replaceAll(DB1_Vocabulary.db1,
    ""))){
                buildComponent=buildComponent+ "qb:Component [qb4o:level
    "+db_name+":"+rLevel+"];"+NL;
                levelExists=true;
                }
            }
        for (int i = 0; i < topos.size(); i++) {
        if (rLevel.equalsIgnoreCase(topos.get(i).replaceAll(DB1_Vocabulary.db1, ""))){
            buildComponent=buildComponent+"qb:Component [igolap:TopoDimension
    "+db_name+":"+rLevel+"];"+NL;
            levelExists=true;
            }
        }
        if (!(levelExists)){return false;}
            }
        return true;
} catch (Exception e) {
    e.printStackTrace();
    System.out.println("Dimensions retrieval exception!");
    return false;
}}
```

Furtheron, the ***calculate()*** method, the same as in the case of the F_Slice operator, it does not address differently the topological and informational members. As a consequence, the query construction phase is the same, building the content for the visualization of the F_Dice operator's output, and the SELECT body for the materialization request from the CONSTRUCT.

```
public static void calculate(String db_name, Model schemasModel, Model
   dataModel,HashMap<String, List<String>> aggLevelMembers,boolean materialize) {
try {
   Model m2 = ModelFactory.createUnion(schemasModel, dataModel);
   String q3 ="";
   String q4Visual ="";
   String q0 = "SELECT DISTINCT ";
   String q0Visual ="";
   String q1 = " ?"+measures.get(0).replaceAll(DB1_Vocabulary.db1, "");
   String q1Constr =" ?id ?"+measures.get(0).replaceAll(DB1_Vocabulary.db1, "");
   String q1a = " WHERE {";
   String q2 =" ?o a qb:Observation;";
   String q0Constr= "";
   String q4Constr="BIND (iri(concat(\""+DB1_Vocabulary.ds+"\","+NL;
   String dsName ="";
   String constructAttr="";
   String rLevel;
   buildComponent=buildComponent+"qb:Component [qb4o:measure " +db_name+":"+
   measures.get(0).replaceAll(DB1_Vocabulary.db1, "")+"]."+NL;
   List <String> rMembers;
   Iterator<String> it = aggLevelMembers.keySet().iterator();
   while (it.hasNext()) {
      rLevel = it.next();
      dsName =dsName+rLevel+"_";
      q0Constr= q0Constr+" ?"+rLevel;
      q0Visual =q0Visual+"?"+rLevel+"_label ";
      rMembers = aggLevelMembers.get(rLevel);
      q1a = q1a + " values ?"+rLevel+" {";
      for (int i = 0; i < rMembers.size(); i++) {
         q1a= q1a+rMembers.get(i)+" ";
      }
      q1a = q1a+"}";
      q2 = q2 +" "+db_name+":"+rLevel+" ?"+rLevel+";";
      q3 = q3 +" ?"+rLevel+" rdfs:label ?"+rLevel+"_label .";
      q4Constr = q4Constr + "STRAFTER(str(?"+rLevel+"), \"#\"),\"_\","+NL;
      constructAttr =constructAttr+db_name+":"+rLevel+" ?"+rLevel+"; ";
      }
      q2=q2+ " "+db_name+":"+measures.get(0).replaceAll(DB1_Vocabulary.db1, "")+"
   ?"+measures.get(0).replaceAll(DB1_Vocabulary.db1, "")+ ". ";
      q4Constr= q4Constr+
   "STRAFTER(str(?"+measures.get(0).replaceAll(DB1_Vocabulary.db1, "")+"),\"#\"))) AS
   ?id)"+NL;
      q4Constr = q4Constr+ "}";
      String q5 =" GROUP BY ?id "+q0Constr+" "+q1;
      q4Visual =q4Visual+"}";
      dsName=dsName+measures.get(0).replaceAll(DB1_Vocabulary.db1, "");
      constructAttr = constructAttr+db_name+":"
   +measures.get(0).replaceAll(DB1_Vocabulary.db1, "")+" ?"
   +measures.get(0).replaceAll(DB1_Vocabulary.db1, "");
      String SelectVisualize = q0+q0Visual+q1+q1a+q2+q3+q4Visual;
      String SelectConstruct= q0+NL+q0Constr+NL+q1Constr+q1a+NL+q2+NL+q4Constr+NL+q5+NL;
```

The body of the built queries is executed as-is for the SELECT request or passed as an input for the CONSTRUCT, based on the value of the materialized parameter, as introduced below:

```
if (materialize) {
   String constructString = prefix + NL + "CONSTRUCT { "+NL + "?id a qb:Observation; " +
   " qb:dataSet ds:dataSet_"+dsName+";"+NL+" "+constructAttr+"}"+NL
                              + " WHERE{ {"
                              + SelectConstruct
                              + "}}";

   Query constructQuery = QueryFactory.create(constructString);
   QueryExecution constr = QueryExecutionFactory.create(constructQuery, m2);
   Model obsModel = constr.execConstruct();
   OutputStream output = new FileOutputStream(dataWrite_path);
```

```
    obsModel.write(output, "N3", null);

    String datasetSchema=buildComponent+"ds:dataSet_"+dsName+" a qb:DataSet;"+NL+"
    qb:structure ds:DailyHhECons." ;
    output.write(datasetSchema.getBytes(Charset.forName("UTF-8")));
    }else {
       String visualizeString = prefix + NL + SelectVisualize+NL ;
       Query queryLevel = QueryFactory.create(visualizeString);
       QueryExecution qe = QueryExecutionFactory.create(queryLevel, m2);
       ResultSet rollup_results = qe.execSelect();
    }
```

6.3.4.2 Exemplification

The F_Dice operator, operates in the same manner on topological dimensions and informational dimensions' members. As such it is not relevant if there are topological or informational members passed to the operator's call.

To reflect this statement, three queries to exemplify the F_Dice operator are introduced below. Two using only one type of members (either informational or topological) and a third one using mixed topological and informational member.

Query 8: Retrieve all consumption observations for Birmingham city, for Months February and March.

Passed parameters to the F_Dice_up operator:

```
aggLevelMembers: {"Month","t:M02Y2011"}, {"Month","t:M03Y2011"}, {"City","l:birm"}
```

Generate SPARQL queries by F_Dice operator for Query 8:

```
SELECT DISTINCT  ?Month_label
   ?City_label ?eCons
 WHERE
 { VALUES ?Month { t:M02Y2011
   t:M03Y2011 }
 VALUES ?City { l:birm }
?o rdf:type qb:Observation .
?o db1:Month ?Month .
?o db1:City ?City .
?o db1:eCons ?eCons .
?Month rdfs:label ?Month_label .
?City rdfs:label ?City_label
}
```

```
CONSTRUCT
 { ?id rdf:type qb:Observation .
?id qb:dataSet ds:dataSet_Month_City_eCons .
?id db1:Month ?Month .
?id db1:City ?City .
?id db1:eCons ?eCons .}
 WHERE
 { { SELECT DISTINCT  ?Month ?City ?id ?eCons
 WHERE
 { VALUES ?Month { t:M02Y2011 t:M03Y2011 }
 VALUES ?City { l:birm }
?o rdf:type qb:Observation .
?o db1:Month ?Month .
?o db1:City ?City .
?o db1:eCons ?eCons
BIND(iri(concat("http://www.energydb.eu/igola
   p/Data/DataSets#", strafter(str(?Month),
   "#"), "_", strafter(str(?City), "#"), "_",
   strafter(str(?eCons), "#"))) AS ?id)
 }
 GROUP BY ?id ?Month ?City ?eCons
 } }
```

Result of the SELECT example:

```
materialize = false;
-------------------------------------------------
| Month label        | City label      | eCons  |
=================================================
| "February 2011@en" | "Birmingham"@en | 140.02 |
| "February 2011@en" | "Birmingham"@en | 185.50 |
| "February 2011@en" | "Birmingham"@en | 231.07 |
| "February 2011@en" | "Birmingham"@en | 395.63 |
| "March 2011@en"    | "Birmingham"@en | 185.90 |
| "March 2011@en"    | "Birmingham"@en | 45.80  |
-------------------------------------------------
```

Result of the CONSTRUCT example:

```
materialize = true;
ds:DailyHhECons a qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [qb4o:measure db1:eCons].

ds:dataSet Month City eCons a qb:DataSet;
 qb:structure ds:DailyHhECons.


ds:M02Y2011_birm_185.50
        a           qb:Observation ;
        qb:dataSet  ds:dataSet Month City eCons ;
        db1:City    l:birm ;
        db1:Month   t:M02Y2011 ;
        db1:eCons   185.50 .

ds:M02Y2011 birm 140.02
        a           qb:Observation ;
        qb:dataSet  ds:dataSet_Month_City_eCons ;
        db1:City    l:birm ;
        db1:Month   t:M02Y2011 ;
        db1:eCons   140.02 .
[..]
```

Query 9: Retrieve all monthly consumption observations for specific households only.

The output is reduced to the available months of February and March, used throughout the exemplification dataset. In this case were selected three Household members as parameters in the operator's call.

```
aggLevelMembers: {"Household"," topo:hh2"},{"Household"," topo:hh119"},{"Household"," 
   topo:hh106"}
```

Generate SPARQL queries by F_Dice operator for Query 9:

```
SELECT DISTINCT  ?Household_label
   ?eCons
 WHERE
 { VALUES ?Household { topo:hh2
   topo:hh106 topo:hh119 }
?o rdf:type qb:Observation .
?o db1:Household ?Household .
?o db1:eCons ?eCons .
?Household rdfs:label
   ?Household_label
 }
```

```
CONSTRUCT
 { ?id rdf:type qb:Observation .
?id qb:dataSet ds:dataSet_Household_eCons .
?id db1:Household ?Household .
?id db1:eCons ?eCons .}
 WHERE
 { { SELECT DISTINCT  ?Household ?id ?eCons
 WHERE
 { VALUES ?Household { topo:hh2 topo:hh106
   topo:hh119 }
?o rdf:type qb:Observation .
?o db1:Household ?Household .
?o db1:eCons ?eCons
BIND(iri(concat("http://www.energydb.eu/igola
   p/Data/DataSets#",
   strafter(str(?Household), "#"), "_",
   str(?eCons))) AS ?id)
 }
 GROUP BY ?id ?Household ?eCons
 } }
```

Result of the SELECT example:

materialize = false;

```
-----------------------------
| Household_label | eCons   |
=============================
| "2"@en          | 185.90 |
| "2"@en          | 231.07 |
| "106"@en        | 140.02 |
| "106"@en        | 45.80  |
| "119"@en        | 31.30  |
| "119"@en        | 395.63 |
-----------------------------
```

Result of the CONSTRUCT example:

materialize = true;

```
ds:DailyHhECons a qb:DataStructureDefinition;
qb:Component [igolap:TopoDimension db1:Household];
qb:Component [qb4o:measure db1:eCons].

ds:dataSet_Household_eCons a qb:DataSet;
 qb:structure ds:DailyHhECons.

ds:hh106 45.80  a      qb:Observation ;
        qb:dataSet     ds:dataSet Household eCons ;
        db1:Household  topo:hh106 ;
        db1:eCons      45.80 .

ds:hh2 185.90  a       qb:Observation ;
        qb:dataSet     ds:dataSet Household eCons ;
        db1:Household  topo:hh2 ;
        db1:eCons      185.90 .

ds:hh119 395.63  a     qb:Observation ;
        qb:dataSet     ds:dataSet Household eCons ;
        db1:Household  topo:hh119 ;
        db1:eCons      395.63 .
```

Query 10: Retrieve all consumption observations for Birmingham city, for Months February and March for two specific households.

Passed parameters to the F_Dice_up operator:

```
aggLevelMembers: {"Month","t:M02Y2011"}, {"Month","t:M03Y2011"}, {"City","l:birm"}
{"Month","t:M02Y2011"}, {"Month","t:M03Y2011"}, {"City","l:birm"}
```

Generate SPARQL queries by F_Dice operator for Query 8:

```
SELECT DISTINCT  ?Month_label
   ?Household_label
   ?City_label ?eCons
 WHERE
 { VALUES ?Month { t:M02Y2011
   t:M03Y2011 }
 VALUES ?Household { topo:hh2
   topo:hh106 topo:hh119 }
 VALUES ?City { l:birm }
?o rdf:type qb:Observation .
?o db1:Month ?Month .
?o db1:Household ?Household .
?o db1:City ?City .
?o db1:eCons ?eCons .
?Month rdfs:label ?Month_label
   .
?Household rdfs:label
   ?Household_label .
?City rdfs:label ?City_label
```

```
CONSTRUCT  { ?id rdf:type qb:Observation .
?id qb:dataset
   ds:dataSet_Month_Household_City_eCons.
?id db1:Month ?Month .
?id db1:Household ?Household .
?id db1:City ?City .
?id db1:eCons ?eCons .}
 WHERE { { SELECT DISTINCT ?Month ?Household ?City
   ?id ?eCons
 WHERE
 { VALUES ?Month { t:M02Y2011 t:M03Y2011 }
 VALUES ?Household {topo:hh2 topo:hh106
   topo:hh119}
 VALUES ?City { l:birm }
?o rdf:type qb:Observation .
?o db1:Month ?Month .
?o db1:Household ?Household .
?o db1:City ?City .
?o db1:eCons ?eCons
BIND(iri(concat("http://www.energydb.eu/igolap/Dat
   a/DataSets#", strafter(str(?Month), "#"), "_",
   strafter(str(?Household), "#"), "_",
   strafter(str(?City), "#"), "_", str(?eCons)))
   AS ?id)
 }
 GROUP BY ?id ?Month ?Household ?City ?eCons }}
```

Result of the SELECT example:

materialize = false;

Month label	Household label	City label	eCons
"February 2011@en"	"2"@en	"Birmingham"@en	231.07
"March 2011@en"	"2"@en	"Birmingham"@en	185.90
"February 2011@en"	"106"@en	"Birmingham"@en	140.02
"March 2011@en"	"106"@en	"Birmingham"@en	45.80
"February 2011@en"	"119"@en	"Birmingham"@en	395.63

Result of the CONSTRUCT example:

materialize = true;

```
ds:DailyHhECons a qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [igolap:TopoDimension db1:Household];
qb:Component [qb4o:level db1:City];
qb:Component [qb4o:measure db1:eCons].
```

```
ds:dataSet_Month_Household_City_eCons a qb:DataSet;
 qb:structure ds:DailyHhECons.

ds:M03Y2011 hh106 birm 45.80
        a               qb:Observation ;
        qb:dataSet      ds:dataSet_Month_Household_City_eCons ;
        db1:City        l:birm ;
        db1:Household   topo:hh106 ;
        db1:Month       t:M03Y2011 ;
        db1:eCons       45.80 .

ds:M02Y2011_hh119_birm_395.63
        a               qb:Observation ;
        qb:dataSet      ds:dataSet Month Household City eCons ;
        db1:City        l:birm ;
        db1:Household   topo:hh119 ;
        db1:Month       t:M02Y2011 ;
        db1:eCons       395.63 .

ds:M03Y2011 hh2 birm 185.90
        a               qb:Observation ;
        qb:dataSet      ds:dataSet_Month_Household_City_eCons ;
        db1:City        l:birm ;
        db1:Household   topo:hh2 ;
        db1:Month       t:M03Y2011 ;
        db1:eCons       185.90 .
```

6.4 Summary

In this chapter, the Federated OLAP operators, which can perform the traditional OLAP aggregations on both informational and topological dimensions of the data were introduced. The three core operators described in this chapter SELECT, CONSTRUCT and MERGE were described for each implementation of the OLAP operators.

Chapter 7 – Evaluation

7.1 Introduction

This chapter illustrates the evaluation of the presented framework. The evaluation criteria are defined to assess the IGOLAP Vocabulary and the Federated OLAP Operators.

In the evaluation process, available benchmarks and comparisons are used. This chapter will be a guide through the evaluation design, data generation, evaluation principles and interpretation of results.

7.2 Evaluation Design and Process

The SPARQL query language is a W3C (W3C Working Group, 2008) recommended querying language for RDF. At the moment there are different benchmarks available for native SPARQL queries and optimisations (Schmidt, Meier, & Lausen, 2010) (Buil-Aranda, Arenas, & Corcho, 2011), SPARQL queries patterns (Perez, Arenas, & Gutierrez, 2009), queries rewriters, and RDF stores (Bizer & Schultz, 2009), but there is no benchmark evaluating OLAP queries performed over data modelled with QB or QB4OLAP (Tennison & TSO, 2011) (Etcheverry & Vaisman, 2011) vocabularies.

As described in Chapter 3 – Methodology, in the evaluation of this work both qualitative and experimental methodological approaches are used.

The provided evaluation is not targeted at providing a benchmark, but will address and highlight the expectations and constraints in regards to performing OLAP operations over Semantic Web data. Further, the interpretation of the results of this evaluation identifies the needed future areas of research, described in Chapter 8.

Although there is no benchmarking for OLAP Operators over the IGOLAP Vocabulary, nor has the vocabulary itself been previously evaluated; the evaluation design considers the evaluation requirements introduced for domain related benchmarks (Grey, 1993). These principles are heavily used in benchmarking processes and furthermore are relevant and applicable to the evaluation of this research as well.

These principles are introduced in Table 7.1, alongside the approach to address these requirements.

Requirement	Description	Evaluation Steps
Relevance	• The testing should take place within a ***specific domain.*** The queries should implement ***realistic requests***. • **Common constellation** of evaluated operators. • **Correctness** of the outputs.	1. Description of the specified domain 2. Design of evaluation queries to emulate realistic requests 3. Identify the relevant operators' characteristic for comparison criteria. 4. Assessment of the operators outcomes in regards to definition and input data
Portability	• Refers to **technical implementation** and ability to execute on different platforms	5. Overview of the limitations of operators' implementation.
Scalability	• The evaluation tests should be run and assessed over small and large datasets (from 10 up to 500 thousand triplets)	6. Description of datasets and definition of datasets incremental step sizes. 7. Description of operators' evaluation results on the provided datasets
Understand ability	• It is highly recommended that queries are kept simple and understandable. • Identification of divers optimisation requirements	8. Introduction of the queries used in the evaluation process and their description 9. Future research topics highlighted based on the evaluation outcome

Table 7.1 Key four evaluation requirements and their addressability

The nine identified evaluation steps are interdependent in the evaluation process and these steps can be executed in a different order. In order to describe these steps, the remaining sections of this chapter are organised as follows:

- ***Specification of the evaluated domain***: highlighting the addressed domain – evaluation step 1.

- ***Evaluation of the queries***: including the introduction of various types of queries and their emulation of real requests, relevant to the identified domain. These queries may make use of different F_Operators, to navigate or retrieve data from different dimensions. – evaluation steps 2, 3, 8

- ***Data overview***: describing available datasets in the evaluation process – evaluation target 6

- ***Federated Operators' evaluation:*** delivering the assessment of the operators, based on their performance, the correctness of their output and constraints. This evaluation is performed on the previously defined set of queries using evaluation steps 3, 4, 5, 7.

- ***Overview of the evaluations***: providing a summary of the performed evaluations and their results and briefly indicating further areas of research. This fulfils the evaluation step 9.

7.3 The Domain for Evaluation

7.3.1 Domain Description

As presented in Chapter 1, the objective of this research is to provide ***OLAP capabilities over the Semantic Web databases*** in order to enhance BI in a given domain. The characteristics, constraints and related work in this area have been described in Chapter 2. This introduced the limitations of the related work in this area, which are addressed through the current work.

7.3.2 Characteristics of the Domain

The operators and vocabulary designed to deliver OLAP capabilities for SW databases, need to support the characteristics of both the Semantic Web and traditional OLAP capabilities. All these characteristics help to define the Vocabulary to model Semantic Web data.

Table 7.2 Multidimensionality modelling concepts in Semantic Web

* (Zhao, Li, Xin, & Han, 2011) does acknowledge the existing of topological structure but treats it as part of the graph network properties

(Etcheverry & Vaisman, 2012) provide a set of the multidimensional modelling features to support the Semantic Web data management, but these are insufficient as it was identified in Chapter 5. Table 7.2 lists a series of works including their identified features and shows the extended characteristics of the IGOLAP Vocabulary to address missing OLAP functionalities.

7.4 Queries for Evaluation

A set of queries that have been designed to assess the operators are presented in this section.

7.4.1 Queries Description

In order to exemplify and evaluate all defined Federated Operators and their capabilities, a compact set of queries were designed to provide a complete evaluation.

The essential criteria for the evaluation of the federated operators:

- Traditional roll-up operation is performed over only informational dimension but the IGOLAP Vocabulary supports two types of dimensions (informational and topological). In order to demonstrate that F_Roll_up can perform over both types, *Query 1 for informational and Query 2 for topological* were designed.
- The aggregation functions in roll-up are evaluated through three queries: *Query 1 for sum, Query 2 for average and Query 3 for count.*
- The Dice operation is performed over dimensions, but does not use aggregation functions. So only two queries *(Query 4 and 5)*, one for each dimension type, are sufficient for validation.
- The Slice operation is performed over only one member at a time. Members can belong to only one dimension type. Consequently two queries (*Queries 6 and 7*) are designed for the two different member's types to evaluate the capabilities of slice operator.
- The Drill operation can be performed over multiple members at one time. Since there are two member types and the order of the requested members is irrelevant, the maximum number of combinations is listed as follows:
    - informational members only – Query 8;
    - topological members only – Query 9;
    - mix of topological and informational members – Query 10.

As presented above, this set of ***minimum ten queries*** offer a ***complete coverage*** of the operators for evaluation.

The ten queries are detailed in Table 3.2 Queries' detailed description and expected outcome introduced in section Steps and methods of the research methodology from Chapter 3.

7.4.2 Emulation of Real World Requests

The queries presented in this section include those designed initially for BI cases on a set of the data provided from the DEHEMS (EU 7th Framework Programme, 2008) research project. The data is based on "Cycle 3" data, which represents the households' electrical consumption from around 250 households in United Kingdom and Bulgaria. The dataset only contains the projects' last 3 months of data: February, March and April.

7.4.3 Queries Aggregation Requirements

Table 7.3 illustrates the required OLAP aggregation and interrogated dimensions for each query, as it was introduced in Chapter 6 – Materialisation of Integrated OLAP Operators for SW Databases.

Query	OLAP Operator	IGOLAP Dimension/Members Type	
		Informational	Topological
Query 1	**Roll-up**	**yes**	**no**
Query 2	**Roll-up**	**no**	**yes**
Query 3	**Roll-up**	**yes**	**no**
Query 4	**Drill**	**yes**	**no**
Query 5	**Drill**	**no**	**yes**
Query 6	**Slice**	**yes**	**no**
Query 7	**Slice**	**no**	**yes**
Query 8	**Dice**	**yes**	**no**
Query 9	**Dice**	**no**	**yes**
Query 10	**Dice**	**yes**	**yes**

Table 7.3 Query mapping to OLAP operators and IGOLAP dimensions type

As it was previously explained, the overview provided in Table 7.2, shows that the previously analysed vocabularies do not have the expressiveness required to model the data queried by the defined set of queries in Table 7.3 above. This is due to missing support for modelling the topological structure as well as some of the required OLAP operators. Providing a parallel comparison between the needed dimensions types and OLAP operations of each query and the QB and QB4OLAP vocabulary capabilities, it can be easily observed which of the queries can not be answered by the previous vocabularies in Table 7.4 as shown below.

Query	**OLAP** Operator	Dimension/Member **Type Requiered**		**QB**	**QB4OLAP**
		Informational	**Topological**	(Tennison & TSO, 2011)	(Etcheverry & Vaisman, 2012)
Query 1	**Roll-up**	**yes**	**no**	Yes	Yes
Query 2	**Roll-up**	**no**	**yes**	**No** (no support for topological structure)	**No** (no support for topological structure)
Query 3	**Roll-up**	**yes**	**no**	Yes	Yes
Query 4	**Drill**	**yes**	**no**	**No** (no de-aggregation concept, no child relationship)	**No** (no de-aggregation concept, no child relationship)
Query 5	**Drill**	**no**	**yes**	**No** (no support for topological structure)	**No** (no support for topological structure)
Query 6	**Slice**	**yes**	**no**	Yes	Yes
Query 7	**Slice**	**no**	**yes**	**No** (no support for topological structure)	**No** (no support for topological structure)
Query 8	**Dice**	**yes**	**no**	**No** (no composition across multiple slices)	Yes
Query 9	**Dice**	**no**	**yes**	**No** (no support for topological structure)	**No** (no support for topological structure)
Query 10	**Dice**	**yes**	**yes**	**No** (no support for topological structure)	**No** (no support for topological structure)

Table 7.4 Queries that cannot be answered by QB or QB4OLAP

7.5 Data Overview

For the evaluation of this work two types of data are used: real datasets and synthetically generated ones. The reason for having synthetic data is to scale-up the size of the dataset for the evaluation of the operators, as the real dataset was not sufficiently large to meet the evaluation criteria.

7.5.1 Real Datasets

As mentioned in the previous subsection, the data used through this work reflects a three month collection of energy consumption data from around 250 households around United Kingdom and Bulgaria. More precisely, data comes from three distinct cities in United Kingdom and two in Bulgaria. This data provides the basis for analytics on regional energy consumption behaviour across different regions inside a country and also between countries.

Due to privacy regulations the households' detailed information cannot be revealed, including the postcodes and users' email addresses. Since it is sensitive data, household income information was difficult to collect, so the data quality was inadequate. In order to compensate for this factor, the linkage between households and income ranges was automatically generated. Nevertheless the initial income ranges values were maintained.

The data was collected in relational database systems and needed to be transformed into Semantic Web data for the purpose of this research work.

7.5.2 Synthetic Dataset

The available data provided two required dataset sizes (10 and 50 thousand triplets). In order to cover the other dataset sizes required in the evaluation process, the data was extrapolated. From the original dataset covering months of February to April, datasets covering the remaining months up to a complete year where achieved.

While the stress performance of the operator is a factor in this evaluation, the focus remains the functionality offered by the introduced vocabulary and operators for OLAP aggregations.

7.6 Federated Operators' Evaluation

As it was introduced in section 7.2 Evaluation Design and Process, in this section all the relevant factors connected to the introduced operators covering: portability, output correctness, performance and constraints are evaluated.

7.6.1 Operators' Portability

The portability of the operators is discussed in regards to three factors:

- ***Programming language of implementation:*** The current operators are implemented in the Java programming language. The version of the Java SDK used is 1.7.

- ***Implementation's libraries used:*** In the implementation of F_Operators the Apache Jena library, which is a Java system for RDF (a RDF API) is used. By using this library, the RDF models can be manipulated directly through Java applications, as well as parsing RDF/XML and N3 RDF serialisations. Additionally the ARQ library is used as a query engine for Jena.

- ***Querying engines for queries execution:*** Currently used in the implementation of the operators and in the evaluation of these, is the ARQ querying engine. This supports SPARQL Queries over RDF data.

Java is a very wide spread and various platforms-compatible programming language for developing applications. Nevertheless, in regards to further portability of this work, the implementation design, logic and complete code is provided and fully explained in Chapter 6. This means that with a minimum of effort the operators can be translated to any programming language that has a library for supporting RDF data manipulation.

In regards to querying engines for the execution of the operators' generated SPARQL queries, there is a broad selection of available querying engines and databases providing endpoints for SPARQL queries. While the evaluation of these solutions is not part of this research work, the addition of a connection to such a query engine or database will have no impact of the current operator's implementation and their

outcome. Nevertheless this can have an impact on the SPARQL query's performance, which is discussed in the operators' performance evaluation section.

7.6.2 Correctness of the Federated Operators' Output

In order to be able to evaluate the correctness of the Federated Operators in this research work, a set of markers have been introduced. These markers are set for each query to examine the actual output of a query against the expected one.

The success criterion requires that a query must pass all the markers. The F_Operator validation requires that all queries using this operator are successfully passed. If any marker of any query is not successfully validated, the entire F_Operator is considered unsuccessful.

The correctness evaluation markers for each query are extracted from the queries described in Table 3.2 Queries' detailed description and expected outcome.

	Marker 1	Marker 2	Marker 3
Q1	Only change in the output data structure is the replacement from component "Day" to "Month" and correctly identified as level.	The new values for the initial measure are reflecting the **arithmetic sum** of contained members in a "Month"	For each member in the "Day" level from the input dataset, a corresponding member for "Month" level is created
Q2	Only change in the output data structure is the replacement from component "Household" to "Income" and correctly identified as topological dimension.	The new values for the initial measure are reflecting the **arithmetic mean** of contained members connected to the same member in the "Income" dimension	For each member in the "Household" dimension from the input dataset, a corresponding member in "Income" dimension is created
Q3	Same Marker 1 of Q1	The new values for the initial measure are reflecting the **counting** of contained members in the "Day" level for represented members in "Month" level.	Same Marker 3 of Q1
Q4	Only change in the output data structure is the replacement from component	The values of the measure reflects existing data on that level	-

	"Month" to "Day" and correctly identified as level.		
Q5	Only change in the output data structure is the replacement from component "Income" to "Household" and correctly identified as topological dimension.	The values of the measure reflects existing data on that level	-

Table 7.5 Identified evaluation markers for Queries 1 to 5

	Marker 1	Marker 2
Q6	Only the "Month" level of the "Time" dimension and the measure remain in the dataset definition and correctly identified as level.	Separate reading coming from different Households, city combination represent separate entries for the "February" member
Q7	Only the "Household" dimension and the measure remain in the dataset definition and correctly identified as topological dimension.	Separate reading coming from different "Month" dimensions, city combination represent separate entries for the "Household" member
Q8	Only the measure and the components "City" and "Month" remain in the data structure definition and are correctly identified as levels.	Only the mentioned members and their values are displayed while combination based on other dimensions represent separate entries for the same unique members
Q9	Same Marker 1 as Q7	Same Marker 2 as in Q8
Q10	Only the requested dimension and the measure remain in the dataset definition and correctly identified as topological dimension or levels in informational dimensions.	Same Marker 2 as in Q8

Table 7.6 Identified evaluation markers for Queries 6 to 10

Since multiple queries make use of the same F_Operator, each operator is evaluated separately against the relevant markers. The required F_Operator for each query was previously introduced in Table 7.3.

The steps for performing this evaluation are introduced in Table 7.7 for the F_Roll_up operator and Appendix C provides the descriptions for the other operators.

Evaluation of F_Operator: F_Roll_up

Query 1 – Marker 1 (Data structure definition in and out)

Data structure from the input dataset vs

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

```
ds:MonthlyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons;
qb4o:hasAggregateFunction qb4o:SUM].
```

output dataset:

The inputted and obtained datasets have the same data structure definition up to the level that was aggregated. The aggregation function used is also written in the new data structure definition, as per defined marker.

Marker Status: **PASSED**

Query 1 – Marker 2

Output values sample for Query 1:

```
"February 2011@en" | "2"@en          | "Birmingham"@en | 231.07
```

Validated against recoded entries for month February of household "2" and the measure's value mathematically proved as arithmetic sum.

```
ds:MonthlyHhECons a qb:DataStructureDefinition;
[..]
qb:Component [qb4o:measure db1:eCons;
qb4o:hasAggregateFunction qb4o:SUM].
```

Marker Status: **PASSED**

Query 1 – Marker 3

Output from the exemplification of the queries contains only 3 distinct months:

Month_label	City_label	Household_label	eCons
"March 2011@en"	"Birmingham"@en	"106"@en	45.80
"March 2011@en"	"Birmingham"@en	"2"@en	185.90
"February 2011@en"	"Birmingham"@en	"106"@en	140.02
"February 2011@en"	"Birmingham"@en	"119"@en	395.63
"February 2011@en"	"Birmingham"@en	"154"@en	185.50
"February 2011@en"	"Birmingham"@en	"2"@en	231.07
"April 2011@en"	"Birmingham"@en	"119"@en	31.30

For each month it was checked that there exists a minimum of 1 day from the specified Month for the specified household.

Marker Status: **PASSED**

Query 2 – Marker 1

Data structure from the input dataset vs output dataset:

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

```
ds:DailyIncomeECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Income];
qb:Component [qb4o:measure db1:eCons;
qb4o:hasAggregateFunction qb4o:AVG].
```

Marker Status: **PASSED**

Query 2 – Marker 2

Output values sample for Query 2:

```
|"Income range 60000 to 70000"@en|"Birmingham"@en|"11th of March 2011@en"| 13.66 |
```

For each income range and day, over all the values of the consumption of households belonging to it, the arithmetic mean was calculated.

Marker Status: **PASSED**

Query 2 – Marker 3

Checked by comparing the query's output with the all income range groups of household and the days that had measures recorded *on the sample dataset*.

Marker Status: **PASSED**

Query 3 – Marker 1

Data structure from the input dataset vs output dataset:

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

```
ds:MonthlyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons;
qb4o:hasAggregateFunction qb4o:COUNT].
```

Marker Status: **PASSED**

Query 3 – Marker 2

For the output of the query over the sample data (***7 entries***), the entries in the initial datasets were manually counted:

```
-------------------------------------------------------------------
| Month_label         | City_label       | Household_label | eCons |
===================================================================
| "March 2011@en"     | "Birmingham"@en  | "106"@en        | 6     |
| "March 2011@en"     | "Birmingham"@en  | "2"@en          | 17    |
| "February 2011@en"  | "Birmingham"@en  | "106"@en        | 22    |
| "February 2011@en"  | "Birmingham"@en  | "119"@en        | 28    |
| "February 2011@en"  | "Birmingham"@en  | "154"@en        | 18    |
| "February 2011@en"  | "Birmingham"@en  | "2"@en          | 15    |
| "April 2011@en"     | "Birmingham"@en  | "119"@en        | 15    |
-------------------------------------------------------------------
```

Marker Status: **PASSED**

Query 3 – Marker 3

For each month it was checked that there exists a minimum of 1 day from the

specified Month for the specified household. **Marker Status**: **PASSED**
Evaluation Result for F_Roll_up: **PASSED**

Table 7.7 F_Roll_up correctness evaluation based on defined markers

The summary of the results of all the operators is available in Table 7.8, while the detailed evaluation steps of performance are provided in Appendix C.

Evaluation of F_Operator: F_Roll_up	*Query 1 – Marker 1*: **PASSED** *Query 1 – Marker 2*: **PASSED** *Query 1 – Marker 3*: **PASSED** *Query 2 – Marker 1*: **PASSED** *Query 2 – Marker 2*: **PASSED** *Query 2 – Marker 3*: **PASSED** *Query 3 – Marker 1*: **PASSED** *Query 3 – Marker 2*: **PASSED** *Query 3 – Marker 3*: **PASSED**
Evaluation of Operator: F_Drill	*Query 4 – Marker 1*: **PASSED** *Query 4 – Marker 2*: **PASSED** *Query 5 – Marker 1*: **PASSED** *Query 5 – Marker 2*: **PASSED**
Evaluation of F_Operator: F_Slice	*Query 6 – Marker 1*: **PASSED** *Query 6 – Marker 2*: **PASSED** *Query 7 – Marker 1*: **PASSED** *Query 7 – Marker 2*: **PASSED**
Evaluation of F_Operator: F_Dice	*Query 8 – Marker 1*: **PASSED** *Query 8 – Marker 2*: **PASSED** *Query 9 – Marker 1*: **PASSED** *Query 9 – Marker 2*: **PASSED** *Query 10 – Marker 1*: **PASSED** *Query 9 – Marker 2*: **PASSED**
F_Operators overall Result: **PASSED**	

Table 7.8 Summary of the outcome of the F_Operators evaluation of correctness

7.6.3 Operators' Performance Evaluation

7.6.3.1 Evaluation Set-up

As previously mentioned there are existing related works in benchmarking SPARQL query performance. A selection of these works (Bizer & Schultz, 2009) (Schmidt, Hornung, Lausen, & Pinkel, 2009) (Schmidt, Meier, & Lausen, 2010) (Buil-Aranda, Arenas, & Corcho, 2011) is used in the evaluation of the test results in the Analysis of the test results section.

The tests are performed under Window 7 Professional machine, Intel Core i7-4800MQ, 2,70 GHz CPU, and 16GB RAM. Data was stored and run from the same

machine where the Eclipse IDE Environment was running. The storage was a 750 GB Western Digital SATA III hard drive, with 16 MB Cache. The Java engines were executed from Eclipse, with JDK1.7.0_79 version.

All the RDF data used in these experiments was in form of N3 serialization of the RDF format.

7.6.3.2 Design and Completion of Operators' Tests

Federated Operators were designed to include a selected number of operators, modifiers and filters specific to the SPARQL querying language. The selection is defined to reduce the complexity and to improve the performance of the operators. The needed SPARQL characteristics in the development of the operators were summarized in Table 6.1, in Chapter 6.

In order to be able to analyse some of the characteristics that influence the performance of the operators, each query was tested twice based on the required output criteria as:

- **a**: the query requests that the output is ***visualised***, so only one SELECT is used.
- **b**: the query requests that the output is ***materialised***, so the CONSTRUCT is used.

The SPARQL characteristics included in a query are relevant in terms of performance. In order to successfully evaluate the selected query set against related benchmarks, the used SPARQL characteristics per query are summarised in Table 7.9.

As the query set is designed for the specified domain, all the queries include the DISTINCT and GROUP modifiers in order to sustain the correctness of the query output and eliminate duplicate solutions.

Table 7.9 presents an overview of the characteristics of each query, and the type of dimensions and members that they operate over.

Characteristic	Select Op.	Construct Op.	Group Mod.	Distinct Mod.	Bind Op.	Values Op.	IRI Op.	Agg. Funct.	Substr operator	Topo Dim.	Info Dim.
Q1a	X		X	X				X			X
Q1b	X	X	X	X	X		X	X	X		X
Q2a	X		X	X				X		X	
Q2b	X	X	X	X	X		X	X	X	X	
Q3a	X		X	X				X			X
Q3b	X	X	X	X	X		X	X	X		X
Q4a	X		X	X							X
Q4b	X	X	X	X	X		X		X		X
Q5a	X		X	X						X	
Q5b	X	X	X	X	X		X		X	X	
Q6a	X		X	X							X
Q6b	X	X	X	X	X		X		X		X
Q7a	X		X	X						X	
Q7b	X	X	X	X	X		X		X	X	
Q8a	X		X	X		X					X
Q8b	X	X	X	X	X	X	X		X		X
Q9a	X		X	X		X				X	
Q9b	X	X	X	X	X	X	X		X	X	
Q10a	X		X	X		X				X	X
Q10b	X	X	X	X	X	X	X		X	X	X

Table 7.9 SPARQL and IGOLAP characteristics per query

This evaluation considers only the SPARQL query language features required by the use case provided. As different benchmarks included different SPARQL characteristics in the queries they use, there is no full match available between this set of queries and the one in the benchmarks available.

The testing of the performance of the operators was designed as follows:

1. *Each query (versions **a** and **b** are tested as two distinct queries) has a designated test.*

2. *There are five input data sizes defined for each of the tests: **10k triplets**, **50k triples**, **150k triplets**, **250k triplets** and **500k triples** with a storage size to up to 15MB each.*

3. *Each test runs 1000 times. The results are used to calculate the mean, the standard deviation and the standard error of each test.*

By designing and conducting the tests in this manner, it gives the possibility of making comparisons among all federated operators, between two operator requests (visualisation or materialisation) and among operators performing over topological dimensions and member and over informational ones.

7.6.4 Analysis of the test results

7.6.4.1 Overview of the results

All the test results discussed in this section are made available in Appendix C.

After obtaining a stable standard deviation and a standard error under 3%, it was concluded that the mean is representative for the performance behaviour of the F_Operators.

The results of the tests are quantified by maximum number of Queries per Second (QpS).

An initial assessment can be realised from the overall maximum number of QpS for each query and set size combination, as illustrated in Figure 7.1 and Figure 7.2.

As previously discussed, the results of the queries are discussed on different group of queries:

- Queries requesting materialisation compared with ones those do not
- Performance of each F_Operator, by analysing the impact of topological and informational dimensions

1. ***Visualising requests vs. materialising requests***

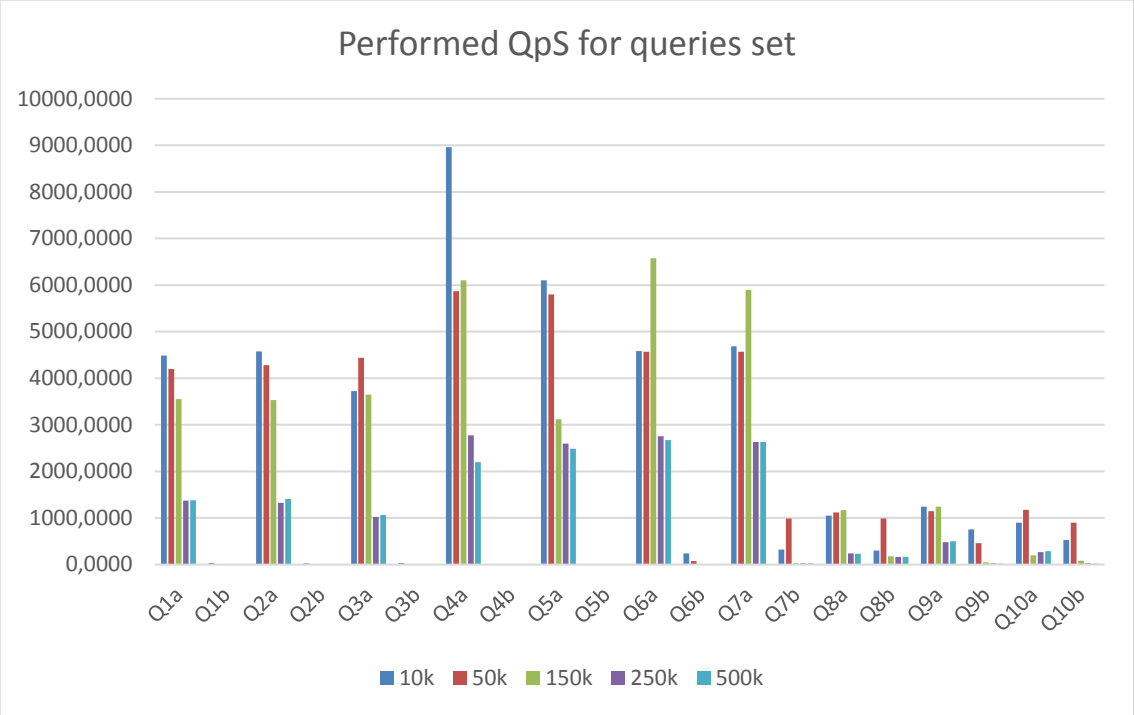


Figure 7.1 Queries overview of achieved QpS 1

The queries materialising the results, and consequently using the CONSTRUCT SPARQL operators, perform consistently worse than the ones without using it.

This is due to the load of the CONSTRUCT SPARQL operator and the additional required modifiers: BIND and IRI over the obtained query.

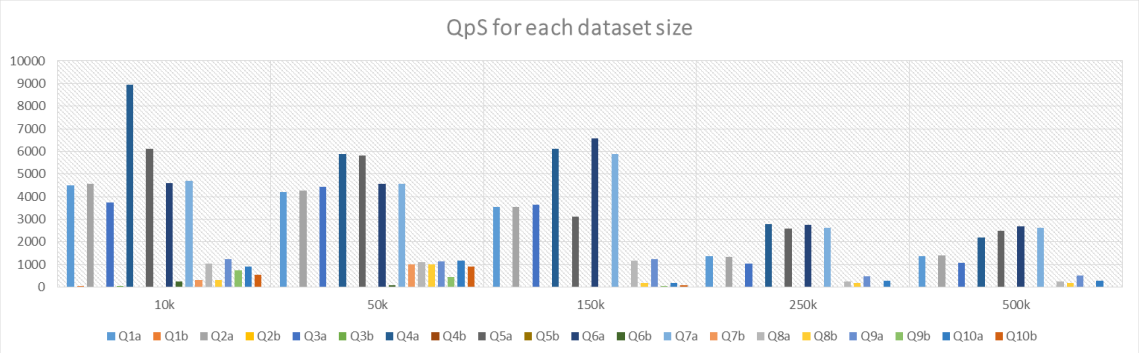


Figure 7.2 Queries overview of achieved QpS 2

Considering that the materialised outputs are further used in other aggregations, gives the possibility of operating on a smaller, pre-aggregated set of data, which can be now be visualised or further aggregated. In terms of performance on a larger volume of data or complex aggregations, the materialised outputs can be used to operate on

reduced (pre-aggregated) datasets (or in steps) and as such optimise the performance of complex requests.

a. Materialising queries

When analysing the construct queries separately, it is observed that all the queries have a linear decrease of performance with the increase in the dataset size as presented in Figure 7.3.

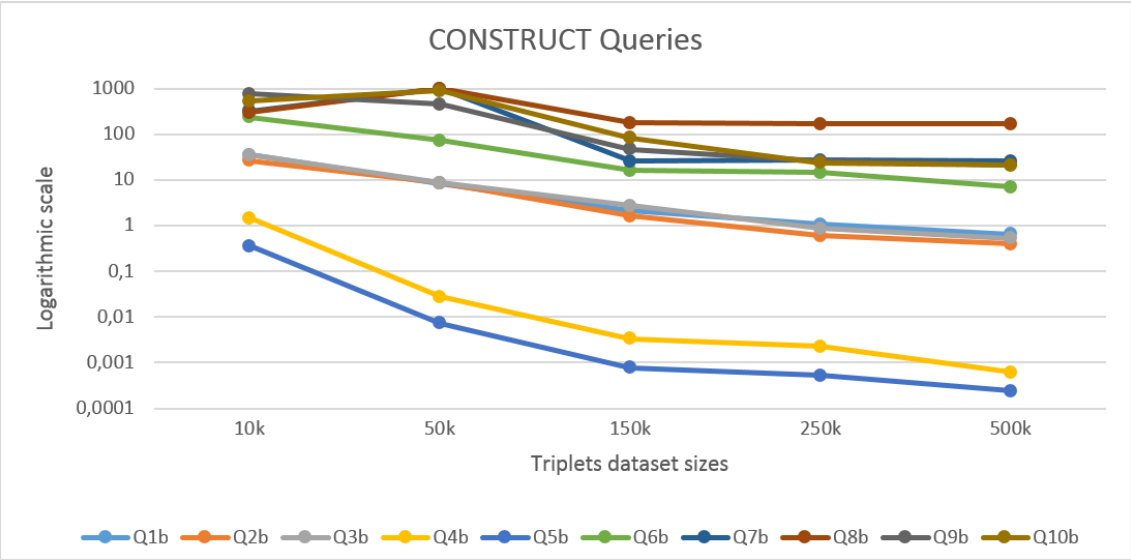


Figure 7.3 Performance across CONSTRUCT queries

The lowest performance was recorded for queries Q4 and Q5 which require the F_Drill operator. The reason behind this low performance is due to the need for calculating a *high number of intermediate responses* and at the same time using the BIND and IRI modifiers.

In Table 7.10 the obtained set QpS averages are summarised while including and excluding the low performing queries from the set.

	Average QpS including Q4 and Q5	Average QpS excluding Q4 and Q5
10k triplets	224.2412463	280.0744499
50k triplets	343.132418	428.9111868
150k triplets	35.42128614	44.27609649
250k triplets	26.0193352	32.52382192
500k triplets	24.29259183	30.36563213

Table 7.10 Average QpS obtained for materialisation requests

The performance of the F_Drill operator (as are all other operators) is detailed in the following paragraphs of this section.

b. Visualising queries

Figure 7.4 illustrates the performance across queries that do not materialise the output. While overall same tendency is registered with regards to decreasing performance with the increase of the dataset size, a distinct pattern is identified here.

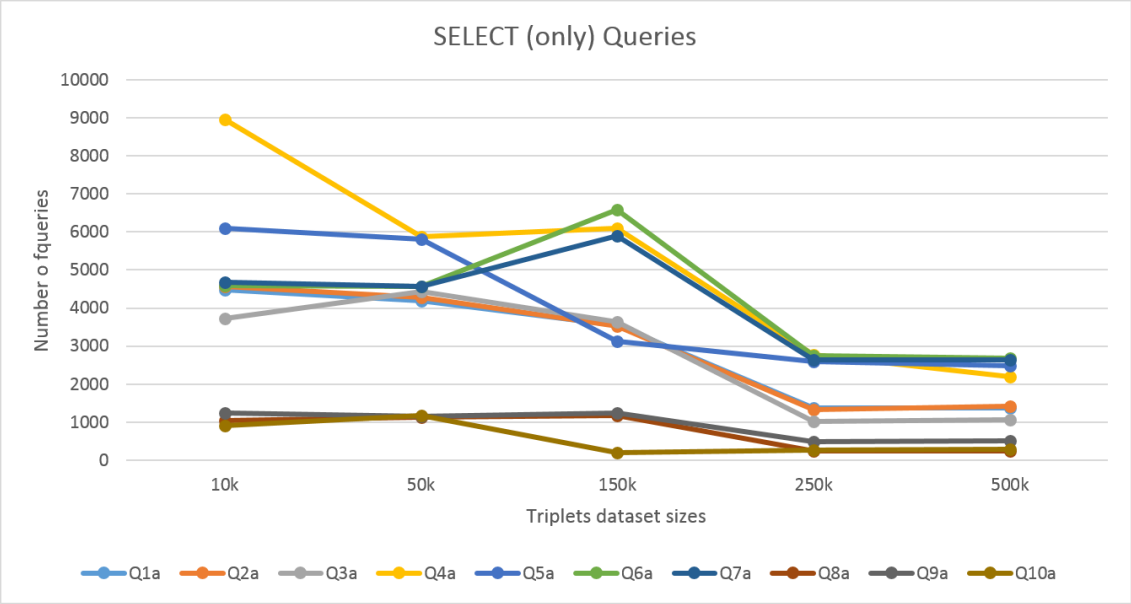


Figure 7.4 Performance across SELECT queries

The lower performance indices in this group are reported for the Q8, Q9 and Q10. All these queries use the F_Dice operator. The F_Dice operator uses the VALUES SPARQL operator in selecting the members needed for the dice operation.

This shows that while the retrieval of intermediate matches in conjunction with the CONSTRUCT, BIND and IRI trigger a high performance decrease, the second most costly SPARQL used operator is the VALUES operator, included in SPARQL 1.1. (W3C Working Group, 2013).

Table 7.11 summarises the obtained average performance between visualising query requests.

	Average QpS including Q8 to Q10	Average QpS excluding Q8 to Q10
10k	4029.967941	5301.952384
50k	3715.395297	4815.86297
150k	3502.979242	4631.163187
250k	1547.21483	2069.053102
500k	1486.523758	1977.083618

Table 7.11 Average QpS obtained for visualisation requests

2. *Federated Operators performance*

a) F_Roll_up

All the queries that are using the F_Roll_up operator have a consistent behaviour. This reflects that *the navigation through the topological or informational dimensions is irrelevant for the performance of the query*.

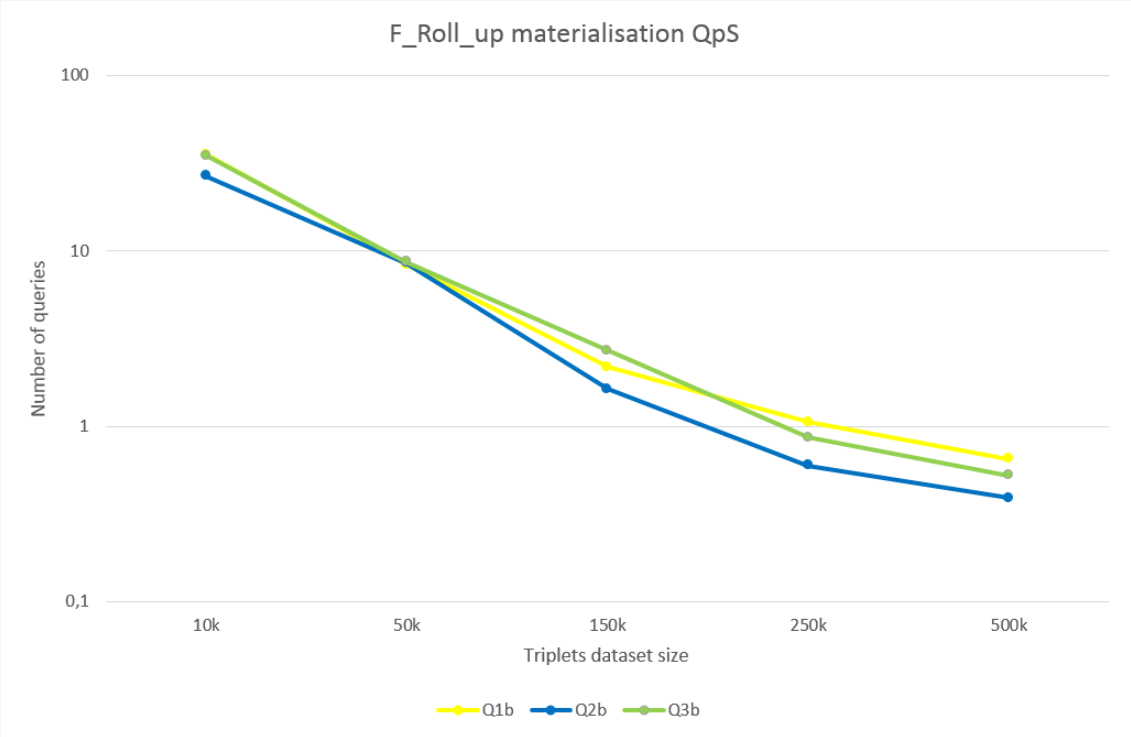


Figure 7.5 F_ROLL_UP visualisation requests

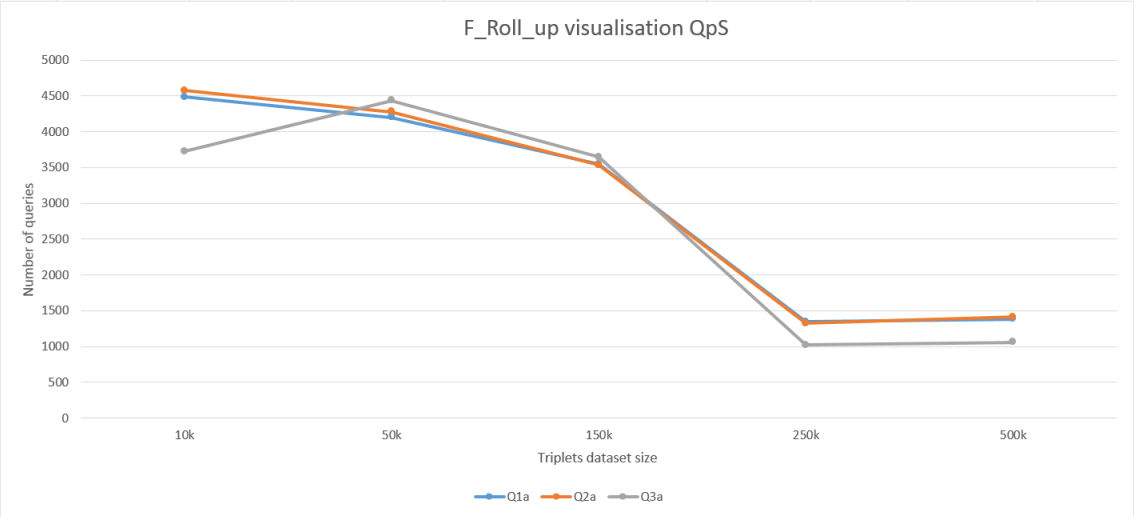


Figure 7.6 F_ROLL_UP materialisation requests

Figure 7.4 and Figure 7.3 show that inside the groups of queries requesting visualisation and respectively materialisation, the F_Roll_up has an average performance in both cases. F_Roll_up supports the usage of SPARQL aggregation functions. This makes us conclude that the usage of SPARQL aggregated functions (SUM, MIN, etc) has a lower impact on the performance than use of the VALUES operator.

b) F_Slice performance

The F_Slice operator queries register the best performance results, both when requesting a materialisation or visualisation only of the outputs.

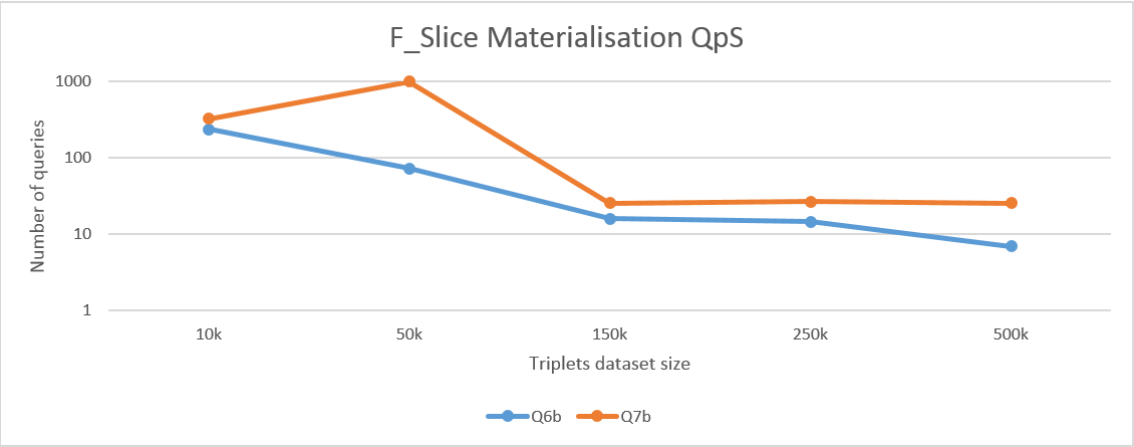


Figure 7.7 F_Slice materialisation queries

The descending pattern of the performance, when the size of the input dataset is increased, it is maintained across the querys using the F_Slice operator. Even in this

case the topological and informational dimensions do not have an impact on the performance of the operator.

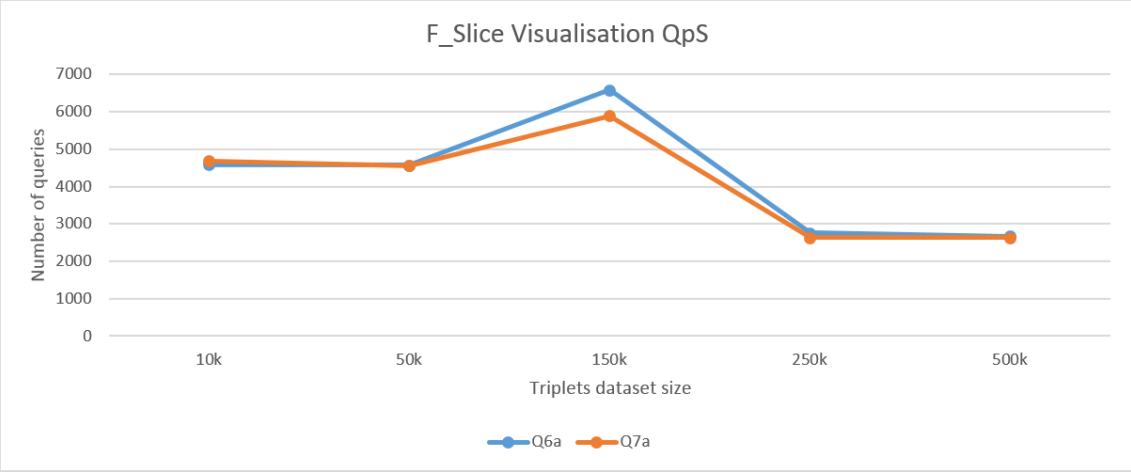


Figure 7.8 F_Slice visualisation queries

c) F_Dice performance

In the case of the F_Dice operator it was identified that this has the lowest performance across visualising operators. This reflects that the use of the VALUES operator it is what mostly affects the performance across SPARQL operators, except for the CONSTRUCT case. Combined with the CONSTRUCT operators, IRI and BIND modifiers show a dramatic downsize of the performance, while the VALUES operator performs better.

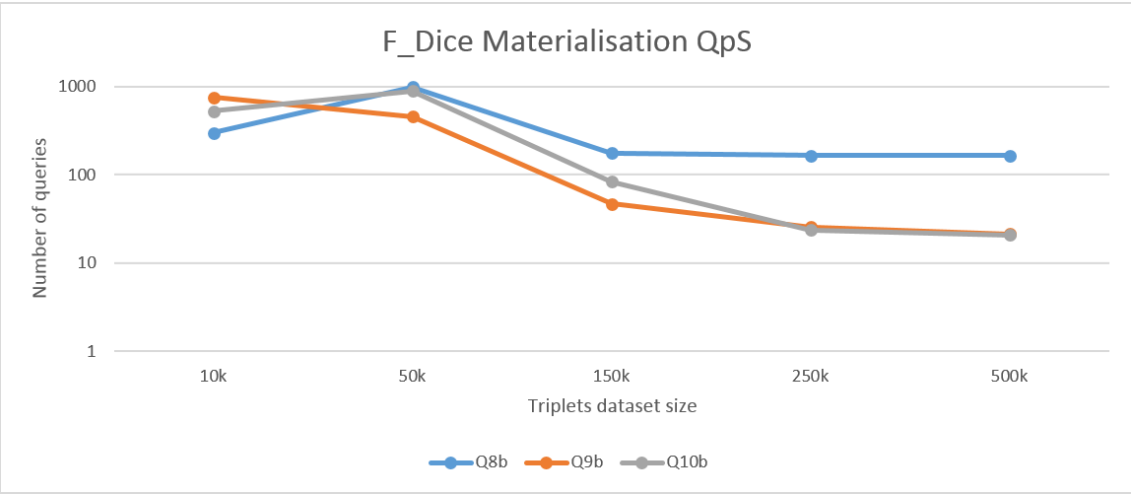


Figure 7.9 F_Dice materialisation queries

Nevertheless, as showed in Figure 7.9 and Figure 7.10, there is no difference between the handling topological or informational members.

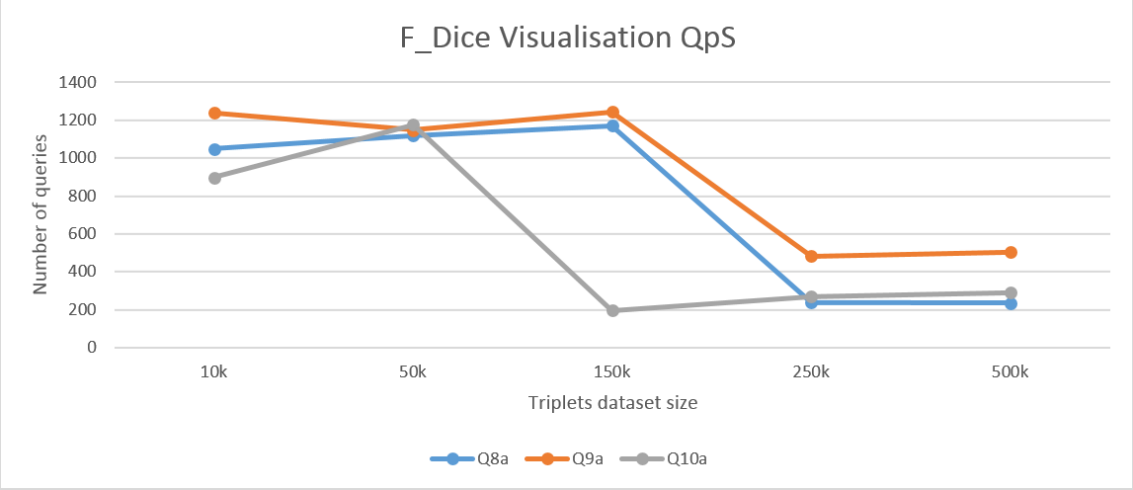


Figure 7.10 F_Dice visualisation queries

d) F_Drill performance

Due to the fact that RDF is organising the data based on a ***Graph Data Model*** and as a consequence the SPARQL querying language is based on ***Statement Patterns (SP)*** which allows it to match individual triples to the ***Graph Pattern*** of the data, the way in which the graph pattern is described highly influences the query performance.

This highlights the fact that the complexity of a query is not only related to a higher number of SPARQL operators and complex filters but also by the order of execution in a query. The reason for this is that the more intermediate results are generated during the execution, the slower the response time of the query becomes.

The later can be exemplified through the F_Drill operator, which has the worst performance throughout the introduced operators. Initially, the Statement Pattern of the SELECT, and as a consequence also the CONSTRUCT operator for the `?subject ?predicate ?object` pattern was implemented as:

```
?Household igolap:topoDConnectedTo ?topoM .
```

Where the ***?Household*** was the ***subject*** with an unknown value and ***?topoM*** was the ***object*** whose value was calculated in one of the previous statements. This would trigger the initial mapping of all available values to ?Household as intermediate results which would then be validated against the matching the ***igolap:topoDConnectedTo*** as the

predicated of the statement and the previously identified values for ***?topoM*** as the ***object*** of the statement.

By changing the above Statement Pattern in the F_Drill operator to the statement introduced below, it was mapped the ***igolap:topoDConnectedTo*** as the ***predicated*** of the statement to the ***?topoM subject***, which is known at this point and as such substantially reducing the number of the matching values for the unknown ***object*** of the statement ***?Household***.

```
?topoM igolap:topoDConnectedTo ?Household.
```

The percent of optimisation on mean, standard deviation and standard error values through this change only is identified for both SELECT and CONSTRUCT in Table 7.12 and respectively Table 7.13 below. The response times in this case are represented in milliseconds and they represent the response time over a 10k triplets dataset.

	Initial Q5a (ms)	Optimised Q5a (ms)	Improvement percent
Mean	0,1907	0,1639	14,05%
Standard Deviation	0,0634	0,0501	20,98%
Standard Error	0,0020	0,0016	21,00%

Table 7.12 Improvement on Query 5 SELECT through SP Optimisation

	Initial Q5b (ms)	Optimised Q5b (ms)	Improvement percent
Mean	3305,495	2819,0893	14,72%
Standard Deviation	67,6434	65,3753	3,35%
Standard Error	2,139	2,0683	3,31%

Table 7.13 Improvement on Query 5 CONSTRUCT through SP Optimisation

After optimisations the tests were retaken for the F_Drill operators and the results obtained are visualised in the below graphics:

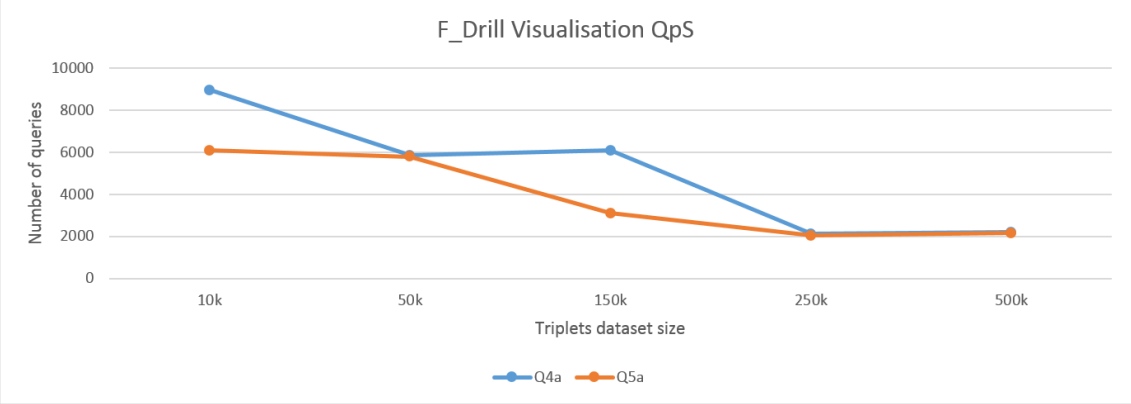


Figure 7.11 F_Drill visualisation requests

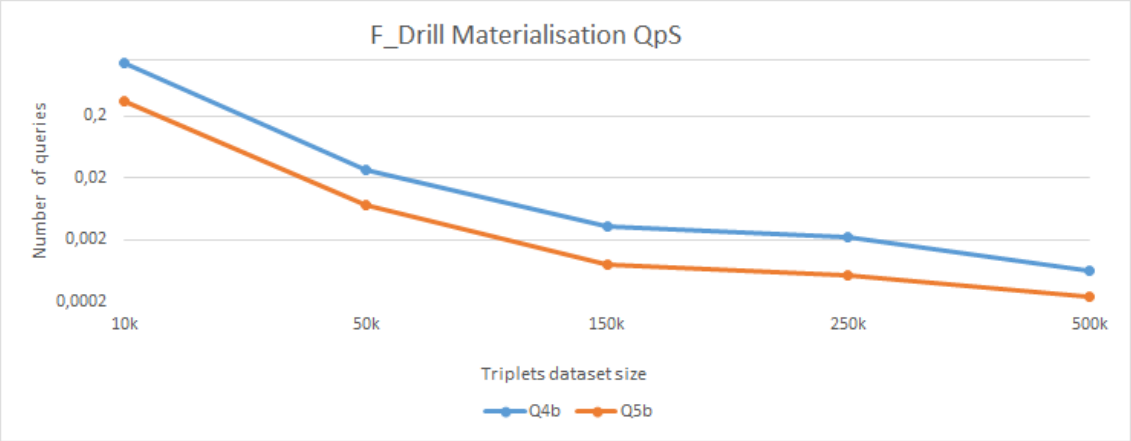


Figure 7.12 F_Drill materialisation requests

Even after the optimisations the ***materialisation option of the F_Drill operators*** has lower outcomes when used in conjunction with the CONSTRUCT SPARQL operator. When the SPARQL CONSTRUCTOR is not used, the BIND and IRI join of operators are implemented. But the visualising option of the F_Dice still uses the VALUES modifier. Due to this, the F_Drill operator performs better than the F_Dice one.

7.6.4.2 Correlation with existing benchmarks (performance aspects)

(Etcheverry & Vaisman, 2012) does not provide any queries evaluation for the introduced vocabulary, as well as no version of visualisation only for the OLAP operators. As a consequence, available relevant benchmarks were selected as reference in this evaluation. (Bizer & Schultz, 2009) address the ***performance*** of SPARQL engines in a prototypical e-commerce scenario. This benchmark do not address

language specific issues, but provides relevant references of expected performance for different complexity e-commerce queries.

The benchmark introduced by (Schmidt, Hornung, Lausen, & Pinkel, 2009), considers additionally to the set of SPARQL specific characteristics in the queries dataset, the performance of in-memory engines. The results of this benchmark are consistent with the results obtained in this work. These results conclude that the performance of the F_Operators is relative to the SPARQL characteristics used.

Furthermore, as shown in (Bizer & Schultz, 2009), available querying engines can respond differently on the same SPARQL operators. The same benchmark includes only one query in their evaluation containing the CONSTRUCT operator, additionally this query does not contain any other modifiers and constraints. This leaves the comparison between the outcomes of this benchmark for the Query 8b (Table 7.14 Comparision between Q8b and Query 12 from (Bizer & Schultz, 2009)). Furthermore the introduced benchmark tests against different native SPARQL engines, while the research tests with an in-memory engine. Nevertheless both of the tested engines are Apache Jena implementations. As shown in table below, the pattern of the query supports a good performance time in comparison with the referenced one.

Query	Querying Engine	Results (QpS)
Query 12 (Bizer & Schultz, 2009)	JenaTDB	53
Q8b	ARQ	123

Table 7.14 Comparision between Q8b and Query 12 from (Bizer & Schultz, 2009)

ARQ –in memory engines (Schmidt, Hornung, Lausen, & Pinkel, 2009) reports on a million triplets dataset, an average performance of across the included tests of queries as **901.73s**, and geometric mean performance of 179.42s. The time out operator (time out time set to 30 minutes) was penalized with 3600s, value that was used to calculate the reported averages.

While still reasonably diverse, the set of queries introduced in this work achieves a mean performance of under **286.96s** on a 500k triplets and a geometrical mean of 0.0321404s on the same dataset size. In this experiment the low performer(s) is not

penalized and the actual performance result of 4115.53s is used in the calculation of the mean.

When referencing to the benchmarks previously introduced, it is shown that the selection of the SPARQL characteristics used in the F_Operators has produced good performance results considering the in-memory engine usage and capability of the SPARQL language.

7.7 Overview of the evaluations

In regards to requirements extracted from (Grey, 1993) which guided the design of this evaluation, the outcome can be summarised as follows:

- **Relevance**: The evaluation is relevant to the OLAP for the Semantic Web domain, and the queries described reflect BI queries from pre-existing project. Evaluation against the results of other existing benchmarks is made only on the common characteristics. Correctness of the outputs is provided in this evaluation against the expected outcomes.

- **Portability:** It is about technical implementation such as libraries, implementation programming language and querying engines.

- **Scalability:** The experiments cover ranges of 10k to 500k triplets input datasets

- **Understability:** It is addressed through deep descriptions of the queries as well as analysis of their outcomes in terms of possible optimisation factors.

It can be observed that the performance of the F_Operators is not influenced by the type of the dimensions (informational or topological) on which they operate.

While performing between the performance-boundaries of their complexity factors, according to the presented benchmarks, the F_Operators are successfully providing OLAP functionality over semantic web data with a lower complexity SPARQL queries. Nevertheless the same tests have to be performed across a broader range of available

querying engines, as these have different performance rates across SPARQL characteristics (Bizer & Schultz, 2009).

Nonetheless, the correctness markers extracted from the expected outcome for the introduced queries were all passed. This shows that the F_Operators achieve a 100% accuracy in regards with their outcome.

All of these conclusions reflect on the capability of the underlying IGOLAP Vocabulary introduced in this work.

7.8 Summary

In this chapter an evaluation of the proposed framework including both IGOLAP Vocabulary and F_Operators was provided. Various aspects of evaluation design and results analysis have been discussed this chapter.

Chapter 8 – Conclusion and future work

8.1 Introduction

The target of this chapter is to summarize in the component sections how the research questions were addressed, what future areas of research were identified and what is the research contribution.

8.2 Research questions coverage

The ***main research question*** was identified in Chapter1 as follows:

> ***Q0 – How can we address Semantic Web data in order to provide OLAP capabilities across distributed SWDBs?***

In order to answer the main research question, a set of relevant questions have been asked and addressed.

- ***Q1 – What do we understand by OLAP over SW data?***

  The answer to this question is delivered in Chapter 2 – Research Background. This chapter contains an introduction into the Semantic Web main concepts, and a discussion of the current research background regarding OLAP in conjunction with the Semantic Web.

- ***Q2 – Why are the current Vocabularies and modelling approaches not suitable to appropriately model SW data for OLAP?***

  The benefits of OLAP capabilities over SW data are illustrated in Chapter 2 and the limitations of existing work in this field was presented in the last subsection of the same chapter: Difficulty in providing OLAP systems over Semantic Web Data. Chapter 4 – presented the proposed integrated approach solution, including a well-defined vocabulary and a set of OLAP operators designed for SW data. Furthermore Chapter 5 – IGOLAP Vocabulary Development introduces the missing capabilities of existing vocabularies and delivers a new vocabulary – IGOLAP – as a solution.

- ***Q3 – How can we perform OLAP over the SW's modelled data?***

The limitations of performing OLAP on SW have been discussed, beginning in Chapter 2, and an overview of the solution to these limitations was presented in Chapter 3. The details regarding architecture and implementation of the IGOLAP vocabulary and Federated Operators, were presented in details in Chapter 5 – IGOLAP Vocabulary Development and Chapter 6 – Materialisation of Integrated OLAP Operators for SW Databases respectively.

- ***Q4 – How will these new set of operators and vocabulary help improve the communication and OLAP capabilities across shared SWDBs?***

  The targeted improvement was introduced in Chapter 1 and was restated and exemplified in Chapter 4 – by presenting real-world case studies. The evaluation of both the vocabulary and operators is provided in Chapter 7 – Evaluation .

By providing answers to these questions, the main research problem was addressed and a complete solution to model and aggregate Semantic Web data by using OLAP operators is achieved.

8.3 Overview of the research contribution

This research contributes to the domain of Semantic Web. It provides an integrated approach to model SW data ***beyond the borders of statistical data*** and enhance BI potentials and OLAP capability through the delivery of a specific vocabulary (IGOLAP) and operators (F_Operators).

Investigation of the research literature has not discovered another available approach that delivers an integrated solution for collective querying over multidimensional semantic web databases. This research makes contributions to the following fields:

- ***Semantic Web data modelling for BI***

  Through the introduction of the Integrated Graph OLAP (IGOLAP) vocabulary for multidimensional data representation, this work contributes to the field of BI specific modelling of the Semantic Web data. The modelling approach targets support for OLAP operators, while considering the graph characteristics of the Semantic Web data. The main characteristics handled by

this vocabulary refers to the informational and topological structures specific to the graph network and as such contained by the Semantic Web data. The relevance of both topological and informational dimensions in graph networks was introduced by the presented related works. Nevertheless this was not previously connected to and applied specifically to Semantic Web data. This aspect was successfully delivered through the IGOLAP Vocabulary. Through the successful introduction and validation of the approach from this work in one comprehensive vocabulary and set of operators, OLAP capabilities were enhanced for the Semantic Web domain.

- ***Data analytics for BI***

  The materialization of a semantic OLAP database capability and delivery of automatically built OLAP operators for SW data provides the necessary support to perform BI data analytics over Semantic Web data.

  The proposed operators are able to provide analytics, in form of different types of aggregations, over both topological and informational dimensions. It was identified that nowadays web data is most likely to be available in topological and mixed topological and informational structures. By introducing these operators it is possible to analyse the web data in conjunction with its statistical counterpart, in order to provide a broader overview and understanding of the semantics of data.

- ***Data integration in the Semantic Web***

  The data generation of new Semantic Web data is produced through the materialisation of the observations generated by the introduced set of operators. These observations are obtained after applying automatically generated SPARQL queries which are applied on the data modelled to the IGOLAP Vocabulary. As a consequence, the generated data conforms with the IGOLAP Vocabulary and allows the publication and reuse of Semantic Web data for further OLAP operations. The obtained Semantic Web data once published with the vocabulary, gives the possibility of integration with other available Semantic Web data, thus enabling further interrogation.

8.4 Foreseen related areas of research

Throughout the evaluation of the current work related areas of research which could improve and extend the research conducted were identified. The main three such research topics are:

1. An evaluation over a more diverse set of Semantic Web data, with a broader mix of both topological and informational dimensions should be conducted, covering both the IGOLAP Vocabulary and F_Operators.

2. A benchmark which includes the performance of the operators across different querying engines and scaled up datasets would provide an even more comprehensive illustration and reference of the constructed work.

3. The optimisation of performance of SPARQL querying language can trigger the improvement of the F_Operators making use of it.

8.5 Reflections

The DEHEMS project (EU 7th Framework Programme, 2008), which is also the first case study of this research work, provided the context of the current work. While set in a traditional BI set-up, early in the project it was observed that the information provided through the semantics of the data is very relevant for deep understanding and analysis of the obtained data. Semantic Web technologies provided the means to describe data, behaviours and relationships, which are very helpful in the reasoning process. By making use of these technologies it would be possible to support standard BI reports with answers to semantics related questions. Additionally it was observed that most of the effort in integrating the OLAP capabilities into the Semantic Web reflected statistical data approaches. These were sustained through a vocabulary designed to model the statistical data in a multidimensional manner. Nevertheless the structure of the Sematic Web is that of an information network graph, where the different edges between the graph nodes relects the relationship between different concepts and thus describes a domain specific data topology.

Additionally, OLAP approaches over graph data delivered the first distinction between the informational and topological dimensions of the information network

graphs. These approaches did not provided any direct applicability to the Semantic Web, nor did they provide a vocabulary to model Semantic Web data for this purpose. Consequently, there was no usage of the standard querying language for the Semantic Web in the description of the generic operators, nor an available implementation and validation of those.

By understanding the benefits that go beyond the usage of these concepts in the context of the Semantic Web, it was possible to provide the missing support and functionality through a defined vocabulary and specifically designed OLAP operators for the newly identified Semantic Web characteristics of the data.

The Semantic Web is starting to become a presence in the Enterprise world. The adoptation of the Semantic Web in the Enterprise IT solutions shows that the conducted research in this area managed to build up an effective set of supporting technologies and approaches. One of the challenges still to be overcome, in order to be fully included in Enterprise solutions, is the gain of trust in BI capabilities. While some results were achieved, there is still work that needs to be done and now is the right time for focusing on these aspects.

References

Agrawal, R., Gupta, A., & Sarawagi, S. (1997). Modeling multidimensional databases. In *Proceedings 13th International Conference on Data Engineering* (pp. 232-243). IEEE.

ArticulateSoftware. (n.d.). *Suggested Upper Merged Ontology (SUMO)*. Retrieved 2010, from http://www.articulatesoftware.com/

Beheshti, S.-M.-R., Benatallah, B., Motahari-Nezhad, H. R., & Allahbakhsh, M. (2012). A framework and a language for on-line analytical processing on graphs. In *Web Information Systems Engineering-WISE 2012* (pp. 213-227). Springer Berlin Heidelberg.

Berlanga, R., Romero Moral, O., Simitsis, A., Nebot, V., Pedersen, T. B., Gamazo, A., & Aramburu, A. (2012). Semantic web technologies for business intelligence. *In Business Intelligence Applications and the Web: Models, Systems and Technologies*, 310-339.

Berners-Lee, T. (1998, February). *Relational Databases on the Semantic Web.* Retrieved from http://www.w3.org/DesignIssues/RDB-RDF.html

Berners-Lee, T. (2006, January). *Design Issues.* Retrieved from http://www.w3.org/DesignIssues/LinkedData.html

Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., . . . Sheets, D. (2006). Tabulator: Exploring and analyzing linked data on the semantic web. Proceedings of the 3rd International Semantic Web User Interaction Workshop.

Berners-Lee, T., Hendler, J., & Lasssila, O. (2001). “The Semantic Web”. *Scientific American Magazine*.

Bizer, C., & Schultz, A. (2009). The Berlin SPARQL Benchmark. *International Journal on Semantic Web & Information, 5*(2).

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data – the story so far. *Int. J. Semantic Web Inf. Syst, 5*(3), 1–22.

Braunschweig, K., Eberius, J., Thiele, M., & Lehner, W. (2012). The State of Open Data Limits of Current Open Data Platforms Categories and Subject Descriptors. Conference Proceedings Eorld Wide Web Conference.

Buil-Aranda, C., Arenas, M., & Corcho, O. (2011). Semantics and optimization of the SPARQL 1.1 federation extension. In S. B. Heidelberg (Ed.), *The Semanic Web: Research and Applications* (pp. 1-15).

Chao, K.-M., Shah, N., Farmer, R., & Matei, A. (2012). Energy Management System for Domestic Electrical Appliances. *International Journal of Applied Logistics (IJAL), 3*(4), 48-60.

Chao, K.-M., Shah, N., Farmer, R., Matei, A., Chen, D.-Y., Schuster-James, H., & Tedd, R. (2010). A profile based energy management system for domestic electrical appliances. In *IEEE 7th International Conference on e-Business Engineering (ICEBE)* (pp. 415-420). IEEE.

Chao, K.-M., Shah, N., Matei, A., Zlamaniec, T., Li, W., Lo, C.-C., & Li, Y. (2011). Intelligent interactive system for collaborative green computing. In *15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 690-697). IEEE.

Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *ACM Sigmod record, 26*(1), 65-74.

Chaudhuri, S., & Dayal, U. (1997). Data warehousing and OLAP for decision support. In *ACM Sigmod Record* (pp. 507-508). ACM.

Chen, C., Yan, X., Zhu, F., Han, J., & Yu, P. S. (2008). Graph OLAP: Towards online analytical processing on graphs. In *Eighth IEEE International Conference on Data Mining, 2008. ICDM'08.* (pp. 103-112). IEEE.

Chen, C., Yan, X., Zhu, F., Han, J., & Yu, P. S. (2009). Graph OLAP: a multi-dimensional framework for graph data analysis. *Knowledge and Information Systems, 21*(1), 41-63.

Codd, E. F. (1993). Beyond decision support. *Computerworld, 27*(30), 87-89.

Cook, T. D., & Reichardt, C. S. (1979). *Qualitative and quantitative methods in evaluation research* (1 ed.). Sage publications Beverly Hills, CA.

d'Aquin, M., & Noy, N. F. (2012). Where to publish and find ontologies? A survey of ontology libraries. *Web Semantics: Science, Services and Agents on the World Wide Web, 11*, 96 - 111.

Elio, R., Hoover, J., Nikolaidis, I., Salavatipour, M., Stewart, L., & Wong, K. (2011). About Computing Science Research Methodology. Retrieved from http://webdocs.cs.ualberta.ca/~c603/readings/research-methods.pdf

Etcheverry, L., & Vaisman, A. (2012). QB4OLAP: a new vocabulary for OLAP cubes on the semantic web. *Proceedings of COLD*.

Etcheverry, L., & Vaisman, A. A. (2011). *QB4OLAP: a Vocabulary for Business Intelligence over the Semantic Web* . Retrieved from http://publishing-multidimensional-data.googlecode.com/git-history/6db60dff91cf4571432f6bf0b31b339579d63795/index.html

Etcheverry, L., & Vaisman, A. A. (2012). Enhancing OLAP analysis with web cubes. In *The Semantic Web: Research and Applications* (pp. 469-483). Heraklion: Springer Berlin Heidelberg.

Etcheverry, L., & Vaisman, A. A. (n.d.). *Open Cube Vocabulary*. Retrieved 2013, from http://purl.org/olap#

EU 7th Framework Programme. (2008). *DEHEMS The Digital Environment Home Energy Management System*. Retrieved 2010, from http://www.dehems.eu/

Fonseca, F., & Martin, J. (2007). Learning the differences between ontologies and conceptual schemas through ontology-driven information systems. *Journal of the Association for Information Systems - Special Issue on Ontologies in the context of IS, 8*(2), 129-142.

Gall, M. D., Borg, W. R., & Gall, J. P. (1996). *Educational research: An introduction* (7 ed.). Longman Publishing.

Greene, J. C., & Caracelli, V. J. (1997). Advances in Mixed-Method Evaluation: The Challenges and Benefits of Integrating Diverse Paradigms. *New Directions for Evaluation, 74*.

Grey, J. (1993). *The Benchmark Handbook for Database and Transaction Systems.* Morgan Kaufmann.

Guarino, N. (1998). *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy.* IOS press.

Heath, T. (2008). The Linking Open Data Project - Bootstrapping the Web of Data. Amsterdam: CATCH Programme and E-Culture Project Meeting on Metadata Interoperability.

Horrocks Ian, K. O. (2006). The even more irresistible SROIQ. *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning*.

Hyland, B., Atemezing, G., & Villazón-Terrazas, B. (2014). *Best Pratices for publishing Linked Data*. Retrieved from World Wide Web Consortium: https://www.w3.org/TR/ld-bp/

Inmon, W. H. (2005). *Building the data warehouse.* John wiley & sons.

Jick, T. D. (1979). Mixing qualitative and quantitative methods:Triangulation in action. *Administrative science quarterly*, 602-611.

Kämpgen, B., & Harth, A. (2011). Transforming statistical linked data for use in OLAP systems. Graz: Proceedings of the 7th international conference on Semantic systems.

Kämpgen, B., O'Riain, S., & Harth, A. (2012). Interacting with statistical linked data via OLAP operations. In *The Semantic Web: ESWC 2012 Satellite Events* (pp. 87-101). Springer Berlin Heidelberg.

Kaplan, B., & Duchon, D. (1988). Combining qualitative and quantitative approaches in information systems research: A case study. *MIS quarterly, 12*, 571-586.

Kaplan, B., & Maxwell, J. A. (2005). Qualitative research methods for evaluating computer information systems. In *Evaluating the organizational impact of healthcare information systems* (pp. 30-55). Springer New York.

Klyne, G., & Carroll, J. (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax - W3C Recommendation.* Retrieved from http://www.w3.org/TR/rdf-concepts/

Kothari, C. R. (2004). *Research methodology: Methods and techniques.* New Age International.

Krötzsch, M., Maier, F., Krisnadhi, A. A., & Hitzler, P. (2011). A Better Uncle For OWL -- Nominal Schemas for Integrating Rules and Ontologies. Hyderabad: WWW 2011 – Session: Query and Ontology Languages.

Le, W., Duan, S., Kementsietsidis, A., Li, F., & Wang, M. (2011). Rewriting queries on SPARQL views. In *Proceedings of the 20th international conference on World wide web* (pp. 655-664). ACM.

Linked Data community. (n.d.). *Linked Data - Connect Distributed Data across the Web*. Retrieved from http://linkeddata.org/

Miles, A., & Bechhofer, S. (2009). *SKOS Simple Knowledge Organization System Namespace Document*. Retrieved from http://www.w3.org/2009/08/skos-reference/skos.html

Morbidoni, C., Polleres, A., Tummarello, G., & Le Phuoc, D. (2007). Semantic web pipes. *Rapport technique, DERI, 71*, 108-112.

Motik, B. (2010). Representing and Querying Validity Time in RDF and OWL: A Logic-based Approach},. In *Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I* (pp. 550-565). Shanghai: Springer-Verlag.

Motik, B., Patel-Schneider, P. F., & Grau, B. C. (2012). *OWL 2 Web Ontology Language Direct Semantics*. Retrieved 2013, from https://www.w3.org/TR/owl2-direct-semantics/

N. Shadbolt, e. a. (2006). The Semantic Web Revised. *IEEE Intelligent Systems, 21*, 96-101.

Noy, N. F., & Mcguinness, D. L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology.* Standford University, Medical Informatics. Standford.

OLAP Council. (1997). *The OLAP glossary.* Retrieved 2013, from http://www.olapcouncil.org

Olive, A. (2004). On the Role of Conceptual Schemas in Information Systems Development. In *Reliable Software Technologies-Ada-Europe 2004* (pp. 16-34). Barcelona: Springer Berlin Heidelberg.

OPEN GOVERNMENT PARTNERSHIP. (2011). *OPEN GOVERNMENT PARTNERSHIP*. Retrieved 2012, from http://www.opengovpartnership.org/about

Orsi, G., & Andreas, P. (2011). Optimizing query answering under ontological constraints. Proc. PVLDB-2011.

OWL Working Group. (2012). *OWL Semantic Web Standards*. Retrieved 2013, from http://www.w3.org/2001/sw/wiki/OWL

Parundekar, R., Knoblock, C. A., & Ambite, J. L. (2010). Linking and Building Ontologies of Linked Data. In *Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part I* (pp. 598--614). Springer-Verlag.

Patton, M. Q. (1990). *Qualitative evaluation and research methods .* SAGE Publications, inc.

Perez, J., Arenas, M., & Gutierrez, C. (2009). Semantics and Complexity of SPARQL. In *ACM Transactions on Database Systems (TODS)* (Vol. 34, p. 16). ACM.

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., & Rosati, R. (2008). Linking Data to Ontologies. In S. Spaccapietra (Ed.), *Journal on Data Semantics X* (Vol. 4900, pp. 133-173). Springer Berlin Heidelberg.

Qu, Q., Zhu, F., Yan, X., Han, J., Philip, S. Y., & Li, H. (2011). Efficient topological OLAP on information networks. In *Database Systems for Advanced Applications* (pp. 389-403). Springer.

Rohloff, K., Dean, M., Emmons, I., Ryder, D., & Sumner, J. (2007). An Evaluation of Triple-Store Technologies for Large Data Stores. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops* (pp. 1105-1114). Springer Berlin Heidelberg.

Schmidt, M., Hornung, T., Lausen, G., & Pinkel, C. (2009). SP2 Bench: A SPARQL Performance Benchmark. In *IEEE 25th International Conference on Data Engineering, 2009. ICDE '09* (pp. 222-233). Shanghai: IEEE.

Schmidt, M., Meier, M., & Lausen, G. (2010). Foundations of SPARQL query optimization. In *Proceedings of the 13th International Conference on Database Theory* (pp. 4-33). ACM.

Shah, N., Chao, K.-M., Zlamaniec, T., & Matei, A. (2011). Ontology for Home Energy Management Domain. In *Proceedings of DICTAP (2)* (pp. 337-347). Springer Berlin Heidelberg.

Statstical Data and Metadata Exchange SDMX. (n.d.). *SDMX 2.1 Technical Specification* . Retrieved 2012, from http://sdmx.org/?page_id=10

Studer, R., Simperl, E., & Kämpgen, B. (2011). Linked Data & Ontologies. Riga, Latvia: STI Summit 2011. Retrieved from http://www.sti-international.org/sites/default/files/Summit%202011/6.%20studer_LDO_v4revRudi.pdf

Tennison, J., & TSO. (2011). *The RDF Data Cube Vocabulary*. Retrieved 2012, from http://www.w3.org/TR/2012/WD-vocab-data-cube-20120405/

The Apache Software Foundation. (2011). *Apache Jena*. Retrieved from https://jena.apache.org/

Tutorial Point. (n.d.). *Data Warehousing Tutorial*. Retrieved 2013, from http://www.tutorialspoint.com/dwh/dwh_olap.htm

Uschold, M. (2011). Making the case for ontology. *Applied Ontology, 6*(4), 377-385.

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *KNOWLEDGE ENGINEERING REVIEW, 11*, 93-136.

Uschold, M., & Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. *ACM SIGMod Record, 33*(4), 58-64.

Vaisman, A., & Zimányi, E. (2014). Data Warehouses and the Semantic Web. In *Data Warehouse Systems* (pp. 539-576). pringer Berlin Heidelberg.

Vassiliadis, P. (1998). Modeling multidimensional databases, cubes and cube operations. In *Proceedings of Tenth International Conference on Scientific and Statistical Database Management, 1998* (pp. 53-62). IEEE.

Vassiliadis, P., & Sellis, T. (1999). A survey of logical models for OLAP databases. *ACM Sigmod Record, 28*(4), 64-69.

Voigt, M., Mitschick, A., & Schulz, J. (2012). Yet Another Triple Store Benchmark? Practical Experiences with Real-World Data. In *SDA* (pp. 85-94). Citeseer.

W3C. (2004, February 10). *OWL Web Ontology Language.* Retrieved 2012, from http://www.w3.org/TR/2004/REC-owl-guide-20040210/#inverseOf

W3C Consortium. (2010, January). *Semantic Web.* Retrieved from http://www.w3.org/standards/semanticweb/

W3C OWL Working Group . (2012). *OWL 2 Web Ontology Language*. Retrieved 2013, from http://www.w3.org/TR/owl2-overview/

W3C Semantic Web. (2004). *OWL Web Ontology Language*. Retrieved 2012, from http://www.w3.org/TR/2004/REC-owl-features-20040210/

W3C Semantic Web. (2008). *Semantic Web Education and Outreach (SWEO) Interest Group*. Retrieved from http://www.w3.org/blog/SWEO/

W3C Semantic Web. (2009). *W3C Semantic Web Frequently Asked Questions*. Retrieved from http://www.w3.org/2001/sw/SW-FAQ

W3C SWEO. (n.d.). *Linking Open Data*. Retrieved 2013, from http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData#Project_Description

W3C Working Group. (2008). *SPARQL Query Language for RDF*. Retrieved 2011, from http://www.w3.org/TR/rdf-sparql-query/

W3C Working Group. (2012). *R2RML: RDB to RDF Mapping Language*. Retrieved 2013, from http://www.w3.org/TR/r2rml/

W3C Working Group. (2013). *SPARQL 1.1 Overview*. Retrieved 2013, from http://www.w3.org/TR/sparql11-overview/

Wenzel, K. (2011). Ontology-Driven Application Architectures with KOMMA,. Bonn: 7th International Workshop on. Retrieved from http://iswc2011.semanticweb.org/

Zhao, P., Li, X., Xin, D., & Han, J. (2011). Graph cube: on warehousing and OLAP multidimensional networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data* (pp. 853-864). ACM.

APPENDIX A - Vocabularies

The RDF Data Cube Vocabulary

(Working draft used at the bases of this research original repository only with the up-to dated versions, version can't be retrieved anymore: http://purl.org/linked-data/cube#
Location of the retrievable version 0.1 of the vocabulary: http://people.aifb.kit.edu/bka/ssb-benchmark/ssb/olap4ld/cube.ttl)

```
@prefix rdf:            <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:           <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:            <http://www.w3.org/2002/07/owl#> .
@prefix xsd:            <http://www.w3.org/2001/XMLSchema#> .
@prefix skos:           <http://www.w3.org/2004/02/skos/core#> .
@prefix foaf:           <http://xmlns.com/foaf/0.1/> .
@prefix scovo:          <http://purl.org/NET/scovo#> .
@prefix void:           <http://rdfs.org/ns/void#> .
@prefix qb:             <http://purl.org/linked-data/cube#> .
@prefix dcterms:        <http://purl.org/dc/terms/> .

<http://purl.org/linked-data/cube>
    a owl:Ontology;
    owl:versionInfo "0.1";
    rdfs:label "The data cube vocabulary";
    rdfs:comment "This vocabulary allows multi-dimensional data, such as statistics, to
   be published in RDF. It is based on the core information model from SDMX (and thus
   also DDI).";
    dcterms:created "2010-07-12"^^xsd:date;
    dcterms:modified "2010-11-27"^^xsd:date;
    dcterms:title "Vocabulary for multi-dimensional (e.g. statistical) data publishing";
    dcterms:license <http://www.opendatacommons.org/licenses/pddl/1.0/> ;
    dcterms:contributor [foaf:mbox "richard@cyganiak.de"],
                        [foaf:mbox "jeni@jenitennison.com"],
                        [foaf:mbox "arofan.gregory@earthlink.net"],
                        [foaf:mbox "ian@epimorphics.com"],
                        [foaf:mbox "dave@epimorphics.com"];
    .

# --- DataSets ---------------------------

qb:DataSet a rdfs:Class, owl:Class;
    rdfs:label "Data set"@en;
    rdfs:comment "Represents a collection of observations, possibly organized into
   various slices, conforming to some common dimensional structure."@en;
    rdfs:subClassOf qb:Attachable;
    owl:equivalentClass scovo:Dataset;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

# --- Observations -----------------------

qb:Observation a rdfs:Class, owl:Class;
    rdfs:label "Observation"@en;
    rdfs:comment "A single observation in the cube, may have one or more associated
   measured values"@en;
    rdfs:subClassOf qb:Attachable;
    owl:equivalentClass scovo:Item;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:dataSet a rdf:Property, owl:ObjectProperty;
    rdfs:label "data set"@en;
    rdfs:comment "indicates the data set of which this observation is a part"@en;
    rdfs:domain qb:Observation;
    rdfs:range qb:DataSet;
    owl:equivalentProperty scovo:dataset;
```

```
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:observation a rdf:Property, owl:ObjectProperty;
    rdfs:label "observation"@en;
    rdfs:comment "indicates a observation contained within this slice of the data
   set"@en;
    rdfs:domain qb:Slice;
    rdfs:range qb:Observation;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

# --- Slices ---------------------------

qb:Slice a rdfs:Class, owl:Class;
    rdfs:label "Slice"@en;
    rdfs:comment "Denotes a subset of a DataSet defined by fixing a subset of the
   dimensional values, component properties on the Slice"@en;
    rdfs:subClassOf qb:Attachable;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:slice a rdf:Property, owl:ObjectProperty;
    rdfs:label "slice"@en;
    rdfs:comment "Indicates a subset of a DataSet defined by fixing a subset of the
   dimensional values"@en;
    rdfs:domain qb:DataSet;
    rdfs:range qb:Slice;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:subSlice a rdf:Property, owl:ObjectProperty;
    rdfs:label "sub slice"@en;
    rdfs:comment "Indicates a narrower slice which has additional fixed dimensional
   values, for example a time-series slice might a subSlice of a slice which spans both
   time and geographic area"@en;
    rdfs:domain qb:Slice;
    rdfs:range qb:Slice;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

# --- Dimensions, Attributes, Measures ---------------------------

qb:Attachable a rdfs:Class, owl:Class;
    rdfs:label "Attachable (abstract)"@en;
    rdfs:comment "Abstract superclass for everything that can have attributes and
   dimensions"@en;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:ComponentProperty a rdfs:Class, owl:Class;
    rdfs:label "Component property (abstract)"@en;
    rdfs:subClassOf rdf:Property;
    rdfs:comment "Abstract super-property of all properties representing dimensions,
   attributes or measures"@en;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .
qb:DimensionProperty a rdfs:Class, owl:Class;
    rdfs:label "Dimension property"@en;
    rdfs:comment "The class of components which represent the dimensions of the
   cube"@en;
    rdfs:subClassOf qb:ComponentProperty;
    rdfs:subClassOf qb:CodedProperty;
    owl:disjointWith qb:MeasureProperty;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .
qb:AttributeProperty a rdfs:Class, owl:Class;
    rdfs:label "Attribute property"@en;
    rdfs:comment "The class of components which represent attributes of observations in
   the cube, e.g. unit of measurement"@en;
    rdfs:subClassOf qb:ComponentProperty;
    owl:disjointWith qb:MeasureProperty;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
```

```
    .
qb:MeasureProperty a rdfs:Class, owl:Class;
    rdfs:label "Measure property"@en;
    rdfs:comment "The class of components which represent the measured value of the
   phenomenon being observed"@en;
    rdfs:subClassOf qb:ComponentProperty;
    owl:disjointWith qb:AttributeProperty;
    owl:disjointWith qb:DimensionProperty;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .
qb:CodedProperty a rdfs:Class, owl:Class;
    rdfs:label "Coded property"@en;
    rdfs:subClassOf qb:ComponentProperty;
    rdfs:comment "Superclass of all coded ComponentProperties"@en;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

# --- Reusable general purpose component properties ---------------------------

qb:measureType a qb:DimensionProperty, rdf:Property;
    rdfs:label "measure type"@en;
    rdfs:comment "Generic measure dimension, the value of this dimension indicates which
   measure (from the set of measures in the DSD) is being given by the obsValue (or
   other primary measure)"@en;
    rdfs:range  qb:MeasureProperty;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

# --- Data Structure Definitions ----------------------------

qb:DataStructureDefinition a rdfs:Class, owl:Class;
    rdfs:label "Data structure definition"@en;
    rdfs:comment "Defines the structure of a DataSet or slice"@en;
    rdfs:subClassOf qb:ComponentSet ;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:structure a rdf:Property, owl:ObjectProperty;
    rdfs:label "structure"@en;
    rdfs:comment "indicates the structure to which this data set conforms"@en;
    rdfs:domain qb:DataSet;
    rdfs:range qb:DataStructureDefinition;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .


qb:component a rdf:Property, owl:ObjectProperty;
    rdfs:label "component specification"@en;
    rdfs:comment "indicates a component specification which is included in the structure
   of the dataset"@en;
    rdfs:domain qb:DataStructureDefinition;
    rdfs:range qb:ComponentSpecification;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

# --- Component specifications - for qualifying component use in a DSD -----------------
   ----------

qb:ComponentSpecification a rdfs:Class, owl:Class ;
    rdfs:label "Component specification"@en;
    rdfs:comment """Used to define properties of a component (attribute, dimension etc)
   which are specific to its usage in a DSD."""@en;
    rdfs:subClassOf qb:ComponentSet ;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:ComponentSet a rdfs:Class, owl:Class;
    rdfs:label "Component set"@en;
    rdfs:comment "Abstract class of things which reference one or more
   ComponentProperties"@en;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .
qb:ComponentSpecification rdfs:subClassOf qb:ComponentSet .
```

```
qb:SliceKey               rdfs:subClassOf qb:ComponentSet .

qb:componentProperty a rdf:Property, owl:ObjectProperty;
    rdfs:label "component"@en;
    rdfs:comment "indicates a ComponentProperty (i.e. attribute/dimension) expected on a
   DataSet, or a dimension fixed in a SliceKey"@en;
    rdfs:domain qb:ComponentSet;
    rdfs:range qb:ComponentProperty;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:order a rdf:Property, owl:DatatypeProperty;
    rdfs:label "order"@en;
    rdfs:comment """indicates a priority order for the components of sets with this
   structure, used to guide presentations - lower order numbers come before higher
   numbers, un-numbered components come last"""@en;
    rdfs:domain qb:ComponentSpecification;
    rdfs:range  xsd:int;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:componentRequired a rdf:Property, owl:DatatypeProperty;
    rdfs:label "component required"@en;
    rdfs:comment """Indicates whether a component property is required (true) or
   optional (false) in the context of a DSD or MSD"""@en;
    rdfs:domain qb:ComponentSpecification;
    rdfs:range  xsd:boolean;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:componentAttachment a rdf:Property;
    rdfs:label "component attachment"@en;
    rdfs:comment """Indicates the level at which the component property should be
   attached, this might an qb:DataSet, qb:Slice or qb:Observation, or a
   qb:MeasureProperty."""@en;
    rdfs:domain qb:ComponentSpecification;
    rdfs:range  rdfs:Class;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:dimension a rdf:Property, owl:ObjectProperty;
    rdfs:label "dimension"@en;
    rdfs:comment "An alternative to qb:componentProperty which makes explicit that the
   component is a dimension"@en;
    rdfs:subPropertyOf qb:componentProperty;
    rdfs:range qb:DimensionProperty;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:measure a rdf:Property, owl:ObjectProperty;
    rdfs:label "measure"@en;
    rdfs:comment "An alternative to qb:componentProperty which makes explicit that the
   component is a measure"@en;
    rdfs:subPropertyOf qb:componentProperty;
    rdfs:range qb:MeasureProperty;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:attribute a rdf:Property, owl:ObjectProperty;
    rdfs:label "attribute"@en;
    rdfs:comment "An alternative to qb:componentProperty which makes explicit that the
   component is a attribute"@en;
    rdfs:subPropertyOf qb:componentProperty;
    rdfs:range qb:AttributeProperty;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:measureDimension a rdf:Property, owl:ObjectProperty;
    rdfs:label "measure dimension"@en;
    rdfs:comment "An alternative to qb:componentProperty which makes explicit that the
   component is a measure dimension"@en;
    rdfs:subPropertyOf qb:componentProperty;
    rdfs:range qb:DimensionProperty;
```

```
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

# --- Slice definitions ----------------------------

qb:SliceKey a rdfs:Class, owl:Class;
    rdfs:label "Slice key"@en;
    rdfs:comment "Denotes a subset of the component properties of a DataSet which are
   fixed in the corresponding slices"@en;
    rdfs:subClassOf qb:ComponentSet ;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:sliceStructure a rdf:Property, owl:ObjectProperty;
    rdfs:label "slice structure"@en;
    rdfs:comment "indicates the sub-key corresponding to this slice"@en;
    rdfs:domain qb:Slice;
    rdfs:range  qb:SliceKey;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:sliceKey a rdf:Property, owl:ObjectProperty;
    rdfs:label "slice key"@en;
    rdfs:comment "indicates a slice key which is used for slices in this dataset"@en;
    rdfs:domain qb:DataSet;
    rdfs:range  qb:SliceKey;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

# --- Concepts ----------------------------

qb:concept a rdf:Property, owl:ObjectProperty;
    rdfs:label "concept"@en;
    rdfs:comment "gives the concept which is being measured or indicated by a
   ComponentProperty"@en;
    rdfs:domain qb:ComponentProperty;
    rdfs:range skos:Concept;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .

qb:codeList a rdf:Property, owl:ObjectProperty;
    rdfs:label "code list"@en;
    rdfs:comment "gives the code list associated with a CodedProperty"@en;
    rdfs:domain qb:CodedProperty;
    rdfs:range skos:ConceptScheme;
    rdfs:isDefinedBy <http://purl.org/linked-data/cube>;
    .
```

The QB4OLAP Vocabulary

(working draft used at the bases of this research original repository: http://purl.org/olap# only with the up-to dated versions, this version can't be retrieved from here: http://purl.org/qb4olap/cubes#

Location of the location of the used version and all versions between, on the versioning control system, publicly available at: https://code.google.com/p/publishing-multidimensional-data/source/browse/rdf/)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix scovo: <http://purl.org/NET/scovo#> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix qb4o: <http://purl.org/olap#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .

<http://purl.org/olap>
a owl:Ontology;
owl:versionInfo "0.1";
rdfs:label "The QB4OLAP cube vocabulary";
rdfs:comment "This vocabulary allows to publish and operate with OLAP cubes in RDF";
dcterms:created "2012-12-06"^^xsd:date;
dcterms:modified "2012-12-06"^^xsd:date;
dcterms:title "Vocabulary for publishing OLAP data cubes";
dcterms:license <http://www.opendatacommons.org/licenses/pddl/1.0/> ;
dcterms:contributor [foaf:mbox "lorenae@fing.edu.uy"],
[foaf:mbox "avaisman@ulb.ac.be"];
.


# --- Levels and Level members ---------------------------

qb4o:LevelProperty a rdfs:Class, owl:Class;
rdfs:label "Level property"@en;
rdfs:comment "The class of components which represent the levels of a dimension"@en;
rdfs:subClassOf qb:ComponentProperty;
rdfs:subClassOf qb:CodedProperty;
rdfs:isDefinedBy <http://purl.org/olap>;
.

qb4o:LevelMember a rdfs:Class, owl:Class;
rdfs:label "Level member"@en;
rdfs:comment "The class of components which represent the members of a level"@en;
rdfs:subClassOf skos:Concept;
rdfs:isDefinedBy <http://purl.org/olap>;
.

qb4o:level a rdf:Property, owl:ObjectProperty;
rdfs:label "level"@en;
rdfs:comment "An alternative to qb:componentProperty which makes explicit that the
   component is a level"@en;
rdfs:subPropertyOf qb:componentProperty;
rdfs:range qb4o:LevelProperty;
rdfs:isDefinedBy <http://purl.org/olap>;
.

qb4o:inDimension a rdf:Property, owl:ObjectProperty;
rdfs:label "level in dimension"@en;
rdfs:comment "Indicates to which dimension the level belongs"@en;
rdfs:range qb4o:LevelProperty;
rdfs:domain qb:DimensionProperty;
```

```
rdfs:isDefinedBy <http://purl.org/olap>;
.

qb4o:inLevel a rdf:Property, owl:ObjectProperty;
rdfs:label "level member in level"@en;
rdfs:comment "Indicates to which level the level member belongs"@en;
rdfs:range qb4o:LevelMember;
rdfs:domain qb4o:LevelProperty;
rdfs:isDefinedBy <http://purl.org/olap>;
.

qb4o:parentLevel a rdf:Property, owl:ObjectProperty;
rdfs:label "is parent of"@en;
rdfs:comment "Indicates which is the parent level of each level"@en;
rdfs:range qb4o:LevelProperty;
rdfs:domain qb4o:LevelProperty;
rdfs:isDefinedBy <http://purl.org/olap>;
.

# --- Aggregate Functions ---------------------------

qb4o:AggregateFunction a rdfs:Class, owl:Class;
rdfs:label "Aggregate function"@en;
rdfs:comment "The class of components which represent aggregate functions that are
   applied to compute measure aggregate values"@en;
rdfs:isDefinedBy <http://purl.org/olap>;
.

qb4o:sum a qb4o:AggregateFunction;
rdfs:label "SUM"@en;
rdfs:comment "Returns the numeric value obtained by adding a set of numeric values."@en;
owl:sameAs dbpedia:Summation;
.

qb4o:avg a qb4o:AggregateFunction;
rdfs:label "AVG"@en;
rdfs:comment "Returns the arithmetic mean of a set of numeric values."@en;
owl:sameAs dbpedia:Average;
.

qb4o:count a qb4o:AggregateFunction;
rdfs:label "COUNT"@en;
rdfs:comment "Returns the number of elements in a set of elements (the cardinality of
   the set)."@en;
owl:sameAs dbpedia:Counting;
.

qb4o:min a qb4o:AggregateFunction;
rdfs:label "MIN"@en;
rdfs:comment "Returns the minimum element in a set of elements, where a partial order is
   defined."@en;
owl:sameAs dbpedia:Min;
.

qb4o:max a qb4o:AggregateFunction;
rdfs:label "MAX"@en;
rdfs:comment "Returns the maximum element in a set of elements, where a partial order is
   defined."@en;
owl:sameAs dbpedia:Max;
.

qb4o:hasAggregateFunction a rdf:Property, owl:ObjectProperty;
rdfs:label "has aggregate function"@en;
rdfs:comment "Indicates which aggregate function has to be applied to obtain measure
   aggregate values, for a certain measure in a cube"@en;
rdfs:range qb:ComponentSpecification;
rdfs:domain qb4o:AggregateFunction;
rdfs:isDefinedBy <http://purl.org/olap>;
.
```

APPENDIX B – Operators' implementation and sample datasets

Source code of Operators

F_Roll_up validate() method:

```
public static boolean validate(String db_name, HashMap<String, String> aggLevelDim, String measure,        String
measureConstraint, Model schemasModel, Model dataModel) {
Iterator<String> it = aggLevelDim.keySet().iterator();
if (it.hasNext()) {
        rLevel = it.next();
        rDim = aggLevelDim.get(rLevel);
        try {
        // retrieve the dimension type of requested to be rolled-up
        String queryDimensionType = prefix + NL + "SELECT ?dimensionType WHERE {" + db_name + ":"+ rDim
+ " a ?dimensionType .}";
        Query queryDimension = QueryFactory.create(queryDimensionType);
        QueryExecution qExe = QueryExecutionFactory.create(queryDimension, schemasModel);
        ResultSet dimType = qExe.execSelect();
        QuerySolution dType = dimType.next();
        dimensionType = dType.get("dimensionType").toString();
        // verify if the dimension is a InfoDimension from the IGOLAP Vocabulary and retrieve its child levels
        if (dimensionType.equalsIgnoreCase(DB1_Vocabulary.igolap + "InfoDimension")) {
                String queryString = prefix + NL + "SELECT ?childLevel WHERE {" + db_name + ":" + rLevel + "
a qb4o:LevelProperty; qb4o:inDimension " + db_name + ":" + rDim + ";igolap:childLevel ?childLevel .}";
                Query query = QueryFactory.create(queryString);
                QueryExecution qe = QueryExecutionFactory.create(query,schemasModel);
                ResultSet results = qe.execSelect();
                while (results.hasNext()) {
                        QuerySolution qs = results.next();
                        String c = qs.get("childLevel").toString();
                        levelChildren.add(c);
                }
                if (levelChildren.isEmpty() || levelChildren == null) {
                        System.out.println("The requiered level doesn't have child levels!");
                        return false;}
                } else {
        // verify if the dimension is a TopoDimension from the IGOLAP Vocabulary and retrieve its connected dom
                if (dimensionType.equalsIgnoreCase(DB1_Vocabulary.igolap + "TopoDimension")) {
                String queryString = prefix     + NL + "SELECT DISTINCT ?topoConnected WHERE
{?topoConnected" + " a igolap:TopoDimension; igolap:topoDConnectedTo "      + db_name + ":" + rDim + " .}";
                Query query = QueryFactory.create(queryString);
                QueryExecution qe = QueryExecutionFactory.create(query,schemasModel);
                ResultSet results = qe.execSelect();
                boolean areConnected = false;
                while (results.hasNext()) {
                        QuerySolution qs = results.next();
                        String c = qs.get("topoConnected").toString().replaceAll(Vocabulary.db1, "");
                        if (c.equalsIgnoreCase(rLevel)) { areConnected = true; }
                        }
                        if (!areConnected) { return areConnected; }
```

```
                }
        }
        // validate that child and measure is in the dataset
        String selectTopoString = prefix + NL + "SELECT DISTINCT ?TopoDims " + "WHERE {?ds a
qb:DataStructureDefinition;" + "?p ?o . ?o igolap:TopoDimension ?TopoDims" + " }";
        Query selectTopoComp = QueryFactory.create(selectTopoString);
        QueryExecution retrieveTopoComp = QueryExecutionFactory.create(selectTopoComp, dataModel);
        ResultSet topoRes = retrieveTopoComp.execSelect();
        while (topoRes.hasNext()) {
                QuerySolution qs = topoRes.next();
                topos.add(qs.get("TopoDims").toString());
        }
        String selectInfoLevelString = prefix + NL          + "SELECT DISTINCT ?InfoLvls "        + "WHERE {?ds
a qb:DataStructureDefinition;" + "?p ?o . ?o qb4o:level ?InfoLvls" + " }";
        Query selectInfoComp = QueryFactory.create(selectInfoLevelString);
        QueryExecution retrieveInfoComp = QueryExecutionFactory.create(selectInfoComp, dataModel);
        ResultSet infoRes = retrieveInfoComp.execSelect();
        while (infoRes.hasNext()) {
                QuerySolution qs = infoRes.next();
                infoLevels.add(qs.get("InfoLvls").toString());
        }
        // validate that dimension, level, measures critarias are fullfiled
        if (dimensionType.equalsIgnoreCase(DB1_Vocabulary.igolap + "InfoDimension")) {
                for (int i = 0; i < infoLevels.size(); i++) {
                        for (int j = 0; j < levelChildren.size(); j++) {
                                if (infoLevels.get(i).equalsIgnoreCase(
                                levelChildren.get(j))) {
                                rlMember = true;
                        }
                }
        }
        } else if (dimensionType.equalsIgnoreCase(DB1_Vocabulary.igolap  + "TopoDimension")) {
                for (int i = 0; i < topos.size(); i++) {
                        if (topos.get(i).equalsIgnoreCase(DB1_Vocabulary.db1 + rDim)) {
                                rTMember = true;
                        }
                }
        }
        String selectMeasureString = prefix + NL + "SELECT ?measure " + "WHERE {?ds a
qb:DataStructureDefinition;" + "?p ?o . ?o qb4o:measure ?measure }";
        Query selectMeasureComp = QueryFactory.create(selectMeasureString);
        QueryExecution retrieveMeasureComp = QueryExecutionFactory.create(selectMeasureComp, dataModel);
        ResultSet measureRes = retrieveMeasureComp.execSelect();
        boolean measure_valid = false;
        while (measureRes.hasNext()) {
                QuerySolution qs = measureRes.next();
                String found_measure = qs.get("measure").toString();
                measures.add(found_measure);
                if (found_measure.equalsIgnoreCase(DB1_Vocabulary.db1 + measure)) {
                        measure_valid = true;}
        }
        String measureConsString = prefix + NL + "SELECT ?measureAgg "+ "WHERE {?measureAgg a
qb4o:AggregateFunction }";
        Query selectMeasureCostr = QueryFactory.create(measureConsString);
        QueryExecution retrieveMeasureCostr = QueryExecutionFactory.create(selectMeasureCostr,
schemasModel);
        ResultSet constrains = retrieveMeasureCostr.execSelect();
        boolean exist_constrain = false;
```

```
            while (constrains.hasNext()) {
                        QuerySolution qs = constrains.next();
                        String found_measure = qs.get("measureAgg").toString();
                        if (found_measure.equalsIgnoreCase(DB1_Vocabulary.qb4o + measureConstraint)) {
                                    exist_constrain = true; }
            }
            if ((measure_valid) && (exist_constrain)&& ((rIMember) || (rTMember))) {
                        return true;
            } else {   return false;}
} catch (Exception e) {
            e.printStackTrace();
            System.out.println("Dimensions retrieval exception!");
            return false;}
} else { System.out.println("No given roll-up target!");
            return false;
}}
```

The entire source code for all F_Operators is available under:

https://www.dropbox.com/l/sh/7u5x9cuyk7mQcoj1TYU2Qo

Sample Datasets

Schemas, dimensions, levels and members definitions of DB1 based on IGOLAP Vocabulary

```
#-Dimensions and Levels definitions of DB1
   based of extended IGOLAP Vocabulary -

db1:hasMaxIncome a owl:DatatypeProperty;
rdfs:label "Miaximum range income" ;
rdfs:domain db1:Income ;
rdfs:range xsd:integer .

db1:hasMinIncome a owl:DatatypeProperty;
rdfs:label "Minimum range income";
rdfs:domain db1:Income ;
rdfs:range xsd:string .

db1:noInhabitants a qb:MeasureProperty ;
rdfs:label "Inhabitants number"@en .

db1:Location a igolap:InfoDimension .

db1:City a qb4o:LevelProperty ;
qb4o:inDimension db1:Location ;
qb4o:parentLevel db1:County .
```

```
db1:Country a qb4o:LevelProperty ;
qb4o:inDimension db1:Location ;
igolap:childLevel db1:Region .

db1:County a qb4o:LevelProperty ;
qb4o:inDimension db1:Location ;
qb4o:parentLevel db1:Region ;
igolap:childLevel db1:City .

db1:Day a qb4o:LevelProperty, xsd:date ;
qb4o:inDimension db1:Time ;
qb4o:parentLevel db1:Month ;
igolap:childLevel db1:Day .

db1:Household a igolap:TopoDimension ;
igolap:topoDConnectedTo db1:Income .

db1:Time a igolap:InfoDimension ;
rdfs:label "dimension of years "@en.

db1:Year a qb4o:LevelProperty,xsd:gYear;
qb4o:inDimension db1:Time ;
```

```
igolap:childLevel db1:Month .

db1:Month a qb4o:LevelProperty,
   xsd:gYearMonth ;
qb4o:inDimension db1:Time ;
qb4o:parentLevel db1:Year ;
igolap:childLevel db1:Day .

db1:Region a qb4o:LevelProperty ;
qb4o:inDimension db1:Location ;
qb4o:parentLevel db1:Country ;
igolap:childLevel db1:County .

db1:Appliance a igolap:TopoDimension ;
rdfs:label "Appliance dimension"@en;
igolap:topoDConnectedTo db1:Household.

db1:Income a igolap:TopoDimension ;
igolap:topoDConnectedTo db1:Appliance,
   db1:Household .
# --- Sample members for the levels of
   dimension Location ---

l:uk a igolap:Member ;
rdfs:label "United Kingdom"@en ;
qb4o:inLevel db1:Country ;
igolap:childLevel l:rWestMid ;
igolap:childLevel l:rSouthWest ;
igolap:childLevel l:rNorthEast .

l:rWestMid a igolap:Member;
rdfs:label "West Midlands Region"@en ;
qb4o:inLevel db1:Region;
igolap:childLevel l:cWestMid ;
qb4o:parentLevel l:uk .

l:cWestMid a igolap:Member;
rdfs:label "West Midlands"@en ;
qb4o:inLevel db1:County ;
igolap:childLevel l:birm ;
qb4o:parentLevel l:rWestMid .

l:birm a igolap:Member;
rdfs:label "Birmingham"@en ;
qb4o:inLevel db1:City ;
qb4o:parentLevel l:cWestMid .

l:rSouthWest a igolap:Member;
rdfs:label "South West Region"@en ;
qb4o:inLevel db1:Region ;
igolap:childLevel l:cBrist ;
qb4o:parentLevel l:uk .

l:cBrist a igolap:Member;
rdfs:label "Bristol Area"@en ;
qb4o:inLevel db1:County ;
igolap:childLevel l:bris ;
qb4o:parentLevel l:rSouthWest .

l:bris a igolap:Member;
rdfs:label "Bristol"@en ;
qb4o:inLevel db1:City ;
qb4o:parentLevel l:cBrist .

l:rNorthEast a igolap:Member;
rdfs:label "North East Region"@en ;
qb4o:inLevel db1:Region;
igolap:childLevel l:cGreaterManc ;
qb4o:parentLevel l:uk .

l:cGreaterManc a igolap:Member;
rdfs:label "Greater Manchester"@en ;
qb4o:inLevel db1:County ;
igolap:childLevel l:manc ;
qb4o:parentLevel l:rNorthEast .

l:manc a igolap:Member;
rdfs:label "Manchester"@en ;
qb4o:inLevel db1:City ;
qb4o:parentLevel l:cGreaterManc .

# --- Sample members of levels in
   dimension Time ---

t:D01M01Y2011 a igolap:Member ;
rdfs:label "1st of January 2011@en";
qb4o:inLevel db1:Day ;
qb4o:parentLevel t:M01Y2011 ;
rdfs:value "2011-01-01"^^xsd:date .

t:D01M02Y2011 a igolap:Member ;
rdfs:label "1st of February 2011@en";
qb4o:inLevel db1:Day ;
qb4o:parentLevel t:M02Y2011 ;
rdfs:value "2011-02-01"^^xsd:date .

t:D01M03Y2011 a igolap:Member ;
rdfs:label "1st of March 2011@en" ;
qb4o:inLevel db1:Day ;
qb4o:parentLevel t:M03Y2011 ;
rdfs:value "2011-03-01"^^xsd:date .
[..]
t:M03Y2011 a igolap:Member ;
rdfs:label "March 2011@en" ;
qb4o:inLevel db1:Month ;
qb4o:parentLevel t:Y2011 ;
igolap:childLevel t:D01M03Y2011,
   t:D02M03Y2011, t:D03M03Y2011,
   t:D04M03Y2011, t:D05M03Y2011,
   t:D06M03Y2011, t:D07M03Y2011,
   t:D08M03Y2011, t:D09M03Y2011,
   t:D10M03Y2011, t:D11M03Y2011,
   t:D12M03Y2011, t:D13M03Y2011,
   t:D14M03Y2011, t:D15M03Y2011,
   t:D16M03Y2011, t:D17M03Y2011,
   t:D18M03Y2011, t:D19M03Y2011,
   t:D20M03Y2011, t:D21M03Y2011,
   t:D22M03Y2011, t:D23M03Y2011,
   t:D24M03Y2011, t:D25M03Y2011,
   t:D26M03Y2011, t:D27M03Y2011,
   t:D28M03Y2011, t:D29M03Y2011,
   t:D30M03Y2011, t:D31M03Y2011 ;
rdfs:value "2011-03"^^xsd:gYearMonth.
[..]
t:Y2011 a igolap:Member ;
rdfs:label "2011@en" ;
qb4o:inLevel db1:Year ;
igolap:childLevel t:M01Y2011, t:M02Y2011,
   t:M03Y2011, t:M04Y2011, t:M05Y2011,
   t:M06Y2011, t:M07Y2011, t:M08Y2011,
   t:M09Y2011, t:M10Y2011, t:M11Y2011,
   t:M12Y2011 ;
rdfs:value "2011"^^xsd:gYear .

# --- Sample members of TopoDimension
   Household and Income triplets ---

topo:hh106 a igolap:Member ;
rdfs:label "106"@en ;
igolap:ofDimension db1:Household ;
igolap:topoDConnectedTo topo:ukIrang3.

topo:hh119 a igolap:Member ;
rdfs:label "119"@en ;
igolap:ofDimension db1:Household ;
igolap:topoDConnectedTo topo:ukIrang2.
```

```
topo:hh154 a igolap:Member ;
rdfs:label "116"@en ;
igolap:ofDimension db1:Household ;
igolap:topoDConnectedTo topo:ukIrang8.
[..]
topo:ukIrang2 a igolap:Member ;
rdfs:label "Income range between 10000 and
   20000"@en ;
igolap:ofDimension db1:Income ;
db1:hasMaxIncome "20000"^^xsd:integer;
db1:hasMinIncome "10000"^^xsd:integer.
```

```
topo:ukIrang3 a igolap:Member ;
rdfs:label "Income range between 20000 and
   30000"@en ;
igolap:ofDimension db1:Income ;
db1:hasMaxIncome "30000"^^xsd:integer;
db1:hasMinIncome "20000"^^xsd:integer.

topo:ukIrang8 a igolap:Member ;
rdfs:label "Income range between 70000 and
   80000"@en ;
igolap:ofDimension db1:Income ;
db1:hasMaxIncome "80000"^^xsd:integer;
db1:hasMinIncome "70000"^^xsd:integer.
```

Dataset containing observations for F_Roll_up exemplification

```
ds:o1 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D08M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "4.01"^^xsd:decimal .

ds:o2 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D09M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "6.73"^^xsd:decimal .

ds:o3 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D19M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "4.94"^^xsd:decimal .

ds:o4 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D22M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "9.30"^^xsd:decimal .

ds:o5 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D17M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "3.85"^^xsd:decimal .

ds:o6 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D04M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "4.83"^^xsd:decimal .

ds:o7 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D18M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.44"^^xsd:decimal .

ds:o8 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D05M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "8.84"^^xsd:decimal .
```

```
ds:o9 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D15M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.44"^^xsd:decimal .

ds:o10 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D06M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "16.22"^^xsd:decimal .

ds:o11 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D16M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.03"^^xsd:decimal .

ds:o12 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D07M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.84"^^xsd:decimal .

ds:o13 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D13M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.60"^^xsd:decimal .

ds:o14 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D14M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "8.21"^^xsd:decimal .

ds:o15 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D01M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.21"^^xsd:decimal .

ds:o16 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D02M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.19"^^xsd:decimal .
```

```
ds:o17 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D11M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "6.05"^^xsd:decimal .

ds:o18 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D03M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.08"^^xsd:decimal .

ds:o19 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D12M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.20"^^xsd:decimal .

ds:o20 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D21M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "6.19"^^xsd:decimal .

ds:o21 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D20M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "3.92"^^xsd:decimal .

ds:o22 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D10M02Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "8.90"^^xsd:decimal .

ds:o23 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D19M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "15.33"^^xsd:decimal .

ds:o24 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D17M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "14.93"^^xsd:decimal .

ds:o25 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D18M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "13.06"^^xsd:decimal .

ds:o26 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D15M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "14.94"^^xsd:decimal .

ds:o27 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D16M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "12.59"^^xsd:decimal .

ds:o28 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D13M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "11.34"^^xsd:decimal .

ds:o29 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D14M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "12.22"^^xsd:decimal .

ds:o30 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D11M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "9.78"^^xsd:decimal .

ds:o31 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D12M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "12.82"^^xsd:decimal .

ds:o32 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D21M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "15.93"^^xsd:decimal .

ds:o33 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D20M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "7.91"^^xsd:decimal .

ds:o34 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D08M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "14.12"^^xsd:decimal .

ds:o35 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D09M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "9.64"^^xsd:decimal .

ds:o36 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D04M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "10.77"^^xsd:decimal .

ds:o37 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D22M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "12.91"^^xsd:decimal .
```

```
ds:o38 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D05M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "12.55"^^xsd:decimal .

ds:o39 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D23M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "12.13"^^xsd:decimal .

ds:o40 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D06M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "12.34"^^xsd:decimal .

ds:o41 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D24M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "16.03"^^xsd:decimal .

ds:o42 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D25M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "18.47"^^xsd:decimal .

ds:o43 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D07M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "17.53"^^xsd:decimal .

ds:o44 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D26M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "13.15"^^xsd:decimal .

ds:o45 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D27M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "14.02"^^xsd:decimal .

ds:o46 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D01M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "20.08"^^xsd:decimal .

ds:o47 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D28M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "16.83"^^xsd:decimal .

ds:o48 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D02M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "20.35"^^xsd:decimal .

ds:o49 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D03M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "14.80"^^xsd:decimal .

ds:o50 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D10M02Y2011;
db1:City l:birm;
db1:Household topo:hh119;
db1:eCons "19.06"^^xsd:decimal .

ds:o51 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D08M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "11.50"^^xsd:decimal .

ds:o52 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D09M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "10.25"^^xsd:decimal .

ds:o53 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D04M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "8.86"^^xsd:decimal .

ds:o54 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D05M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "11.16"^^xsd:decimal .

ds:o55 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D06M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "12.02"^^xsd:decimal .

ds:o56 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D25M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "10.96"^^xsd:decimal .

ds:o57 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D07M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "10.81"^^xsd:decimal .

ds:o58 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D26M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
```

```
db1:eCons "9.46"^^xsd:decimal .

ds:o59 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D13M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "14.88"^^xsd:decimal .

ds:o60 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D27M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "6.64"^^xsd:decimal .

ds:o61 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D14M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "9.02"^^xsd:decimal .

ds:o62 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D01M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "8.83"^^xsd:decimal .

ds:o63 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D28M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "5.65"^^xsd:decimal .

ds:o64 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D02M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "13.67"^^xsd:decimal .

ds:o65 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D11M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "10.14"^^xsd:decimal .

ds:o66 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D03M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "9.80"^^xsd:decimal .
```

```
ds:o67 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D12M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "9.87"^^xsd:decimal .

ds:o68 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D10M02Y2011;
db1:City l:birm;
db1:Household topo:hh154;
db1:eCons "11.98"^^xsd:decimal .

ds:o69 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D04M03Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "4.83"^^xsd:decimal .

ds:o70 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D18M03Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.44"^^xsd:decimal .

ds:o80 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D05M03Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "8.84"^^xsd:decimal .

ds:o90 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D15M03Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.44"^^xsd:decimal .

ds:o100 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D06M03Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "16.22"^^xsd:decimal .

ds:o110 a qb:Observation;
qb:dataSet ds:dataSet-uk;
db1:Day t:D16M03Y2011;
db1:City l:birm;
db1:Household topo:hh106;
db1:eCons "5.03
```

APPENDIX C – Results of the evaluation

Evaluation of correctness of all F_Operators

The evaluation of correctness it is perfomed against all defined markers for all operators. The evaluation and its results are provided for each operator below.

1. Evaluation of F_Operator: F_Roll_up:

Query 1 – Marker 1 – **Marker Status**: **PASSED**

Data structure from the input dataset vs.

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

output dataset:

Query 1 – Marker 2 – **Marker Status**: **PASSED**

Output values sample for Query 1:

```
"February 2011@en" | "2"@en          | "Birmingham"@en | 231.07
```

Validated against recoded entries for month February of household "2" and the measure's value mathematically proved as arithmetic sum.

Query 1 – Marker 3

Output from the exemplification of the queries contains only 3 distinct months:

Month_label	City_label	Household_label	eCons
"March 2011@en"	"Birmingham"@en	"106"@en	45.80
"March 2011@en"	"Birmingham"@en	"2"@en	185.90
"February 2011@en"	"Birmingham"@en	"106"@en	140.02
"February 2011@en"	"Birmingham"@en	"119"@en	395.63
"February 2011@en"	"Birmingham"@en	"154"@en	185.50
"February 2011@en"	"Birmingham"@en	"2"@en	231.07
"April 2011@en"	"Birmingham"@en	"119"@en	31.30

For each month it was checked that exists minimum 1 day from the specified Month for the specified household.

Query 2 – Marker 1 – **Marker Status**: **PASSED**

Data structure from the input dataset vs output dataset:

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

Query 2 – Marker 2 – **Marker Status**: **PASSED**

Output values sample for Query 2:

```
|"Income range 60000 to 70000"@en|"Birmingham"@en|"11th of March 2011@en"| 13.66 |
```

For each income range and day, over all the values of the consumption of households belonging to it was calculated the arithmetic mean.

Query 2 – Marker 3 **Marker Status**: **PASSED**

Checked by comparing the query's output with the all income range groups of household and the days that these had measures recorded on the sample dataset.

Query 3 – Marker 1 **Marker Status**: **PASSED**

Data structure from the input dataset vs

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

output dataset:

Query 3 – Marker 2 – **Marker Status**: **PASSED**

For the output of the query over the sample data (7 entries), it was manually counted the entries in the initial datasets:

```
-------------------------------------------------------------------
| Month_label         | City_label       | Household_label | eCons |
===================================================================
| "March 2011@en"     | "Birmingham"@en | "106"@en        | 6     |
| "March 2011@en"     | "Birmingham"@en | "2"@en          | 17    |
| "February 2011@en" | "Birmingham"@en | "106"@en        | 22    |
| "February 2011@en" | "Birmingham"@en | "119"@en        | 28    |
| "February 2011@en" | "Birmingham"@en | "154"@en        | 18    |
| "February 2011@en" | "Birmingham"@en | "2"@en          | 15    |
| "April 2011@en"     | "Birmingham"@en | "119"@en        | 15    |
-------------------------------------------------------------------
```

Query 3 – Marker 3 – **Marker Status**: **PASSED**

For each month it was checked that exists minimum 1 day from the specified Month for the specified household.

Evaluation Result for F_Roll_up: **PASSED**

2. Evaluation of F_Operator: F_Drill:

Query 4 – Marker 1 – **Marker Status**: **PASSED**

Data structure from the input dataset vs

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

output dataset:

Query 4 – Marker 2 – **Marker Status**: **PASSED**

Output values sample for Query 4:

```
| "2nd of February 2011@en"  | "Birmingham"@en | "2"@en  | 16.67 |
```

Validated against recoded entries for month February of household "2" that there is minimum a day entry in the output and the values of the measure reflects existing data on that level.

Query 5 – Marker 1 – **Marker Status**: **PASSED**

Data structure from the input dataset vs output dataset:

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Day];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure
   db1:eCons].
```

Query 5 – Marker 2 – **Marker Status**: **PASSED**

Output values sample for Query 5:

```
| "2"@en          | "17th of March 2011@en"    | "Birmingham"@en | 12.89 |
```

The values of the measure reflects existing data on that dimension.

Evaluation Result for F_Drill: **PASSED**

3. Evaluation of F_Operator: F_Slice:

Query 6 – Marker 1 – **Marker Status**: **PASSED**

Data structure from the input dataset vs

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure
   db1:eCons].
```

output dataset:

Query 6 – Marker 2 – **Marker Status**: **PASSED**

Output values sample for Query 6:

```
| requested Month    | City label      | Household label | eCons  |
=====================================================================
| "February 2011@en" | "Birmingham"@en | "2"@en          | 231.07 |
| "February 2011@en" | "Birmingham"@en | "119"@en        | 395.63 |
| "February 2011@en" | "Birmingham"@en | "106"@en        | 140.02 |
| "February 2011@en" | "Birmingham"@en | "154"@en        | 185.50 |
---------------------------------------------------------------------
```

Validated that all the retrieved values are only for the requested month level in Time dimension - February.

Query 7 – Marker 1 – **Marker Status**: **PASSED**

Data structure from the input dataset vs output dataset:

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

Query 7 – Marker 2 – **Marker Status**: **PASSED**

Output values sample for Query 7:

```
-------------------------------------------------------------------------------
| requested_Household | City_label      | Month_label        | eCons  |
===============================================================================
| "119"@en            | "Birmingham"@en | "February 2011@en" | 395.63 |
| "119"@en            | "Birmingham"@en | "April 2011@en"    | 31.30  |
-------------------------------------------------------------------------------
```

Validated that all the retrieved member are only the requested: specific household.

Evaluation Result for F_Slice: **PASSED**

4. Evaluation of F_Operator: F_Dice:

Query 8 – Marker 1 – **Marker Status**: **PASSED**

Data structure from the input dataset vs output dataset:

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

Query 8 – Marker 2 – **Marker Status**: **PASSED**

Output values sample for Query 1:

```
-------------------------------------------------
| Month label         | City label      | eCons  |
=================================================
| "February 2011@en" | "Birmingham"@en | 140.02 |
| "February 2011@en" | "Birmingham"@en | 185.50 |
| "February 2011@en" | "Birmingham"@en | 231.07 |
| "February 2011@en" | "Birmingham"@en | 395.63 |
| "March 2011@en"    | "Birmingham"@en | 185.90 |
| "March 2011@en"    | "Birmingham"@en | 45.80  |
```

Validated against recoded entries that all outcomes are as requested per members:

```
{"Month","t:M02Y2011"}, {"Month","t:M03Y2011"}, {"City","l:birm"}
```

Query 9 – Marker 1 – **Marker Status**: **PASSED**

Data structure from the input dataset vs output dataset:

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons].
```

Query 9 – Marker 2 – **Marker Status**: **PASSED**

Output values sample for Query 9:

```
----------------------------
| Household_label | eCons  |
============================
| "2"@en          | 185.90 |
| "2"@en          | 231.07 |
| "106"@en        | 140.02 |
| "106"@en        | 45.80  |
```

For each members returned it is checked that are as requested by input parameters:

```
{"Household"," topo:hh2"},{"Household"," topo:hh119"},{"Household"," topo:hh106"}
```

Query 10 – Marker 1 – **Marker Status**: **PASSED**

Data structure from the input dataset vs

```
ds:DailyHhECons a
   qb:DataStructureDefinition;
qb:Component [qb4o:level db1:Month];
qb:Component [qb4o:level db1:City];
qb:Component [igolap:TopoDimension
   db1:Household];
qb:Component [qb4o:measure db1:eCons;
qb4o:hasAggregateFunction qb4o:SUM].
```

output dataset:

Query 10 – Marker 2 – **Marker Status**: **PASSED**

Output from query 10:

Month_label	Household_label	City_label	eCons
"February 2011@en"	"2"@en	"Birmingham"@en	231.07
"March 2011@en"	"2"@en	"Birmingham"@en	185.90
"February 2011@en"	"106"@en	"Birmingham"@en	140.02
"March 2011@en"	"106"@en	"Birmingham"@en	45.80
"February 2011@en"	"119"@en	"Birmingham"@en	395.63

The output is checked that all members belong to the input parameter members:

```
{"Month","t:M02Y2011"}, {"Month","t:M03Y2011"}, {"City","l:birm"}
{"Month","t:M02Y2011"}, {"Month","t:M03Y2011"}, {"City","l:birm"}
```

Evaluation Result for F_Dice: **PASSED**

Additional evaluation results

In thise section of Appendix C are provided additional results of the queries performed, in summarised tables and figures.

1. **Mean, standard deviation and standard error values from 1000 calls of each from the 10 queries summariesd in Tables 1 and 2.**

		Q1a	Q1b	Q2a	Q2b	Q3a	Q3b	Q4a	Q4b	Q5a	Q5b
10k triplets	Mean	0.00022	0.02839	0.00022	0.03740	0.00027	0.02862	0.00011	0.68393	0.00016	2.81909
	STDEV	0.00018	0.00626	0.00018	0.01056	0.00042	0.00627	0.00004	0.02276	0.00005	0.06623
	Standard Error	0.00001	0.00020	0.00001	0.00033	0.00001	0.00020	0.00000	0.00072	0.00000	0.00209
50k triplets	MEAN	0.00024	0.12013	0.00023	0.11766	0.00023	0.11548	0.00017	36.48009	0.00017	137.48772
	STDEV	0.00012	0.00710	0.00012	0.00718	0.00011	0.00823	0.00005	4.58260	0.00005	1.25042
	Standard Error	0.00000	0.00022	0.00000	0.00023	0.00000	0.00026	0.00000	0.32650	0.00000	0.32286
150k triplets	Mean	0.00028	0.45830	0.00028	0.61250	0.00027	0.36983	0.00016	302.33809	0.00032	1278.90939
	STDEV	0.00014	0.01580	0.00014	0.04165	0.00014	0.03508	0.00001	0.63105	0.00002	0.00000
	Standard Error	0.00000	0.00050	0.00000	0.01317	0.00000	0.01109	0.00000	0.19955	0.00001	0.00000

Table 1 - Mean, Standard deviation and Standard Error for Q1 to Q5 on 10k, 50k and 150k triplets datasets

		Q6a	Q6b	Q7a	Q7b	Q8a	Q8b	Q9a	Q9b	Q10a	Q10b
10k triplets	Mean	0.00022	0.00421	0.00021	0.00309	0.00095	0.00333	0.00081	0.00133	0.00111	0.00190
	STDEV	0.00007	0.00153	0.00006	0.00079	0.00090	0.00224	0.00073	0.00071	0.00117	0.00148
	Standard Error	0.00000	0.00005	0.00000	0.00002	0.00003	0.00007	0.00002	0.00002	0.00004	0.00005
50k triplets	MEAN	0.00022	0.01372	0.00022	0.00101	0.00089	0.00101	0.00087	0.00219	0.00085	0.00112
	STDEV	0.00005	0.00285	0.00005	0.00052	0.00088	0.00052	0.00050	0.00060	0.00063	0.00074
	Standard Error	0.00000	0.00009	0.00000	0.00002	0.00003	0.00002	0.00002	0.00002	0.00002	0.00002
150k triplets	Mean	0.00015	0.06272	0.00017	0.03904	0.00085	0.00570	0.00080	0.02121	0.00509	0.01196
	STDEV	0.00000	0.03583	0.00001	0.00324	0.00062	0.00064	0.00037	0.00579	0.00063	0.00266
	Standard Error	0.00000	0.01133	0.00000	0.00102	0.00002	0.00020	0.00001	0.00183	0.00020	0.00084

Table 2 - Mean, Standard deviation and Standard Error for Q1 to Q5 on 10k, 50k and 150k triplets datasets

2. **Mean, standard deviation and standard error of the F_Drill operator over topological dimension, with visualisation only - Q5a – and additionally with materialisation - Q5b .**

	Q5a - initial	Q5a - optimised	Q5b - initial	Q5b - optimised
	Millisec.	Millisec.	Millisec.	Millisec.
Mean	0.19076	0.16390	3305.49504	2819.08939
STDEV	0.06342	0.05013	67.64348	65.37533
Standard Error	0.00200	0.00158	2.13907	2.06838

Table 3 - Collected results tests on Query 5a and 5 b before and after SP Optimisation on a set of 65k triplets

3. **Average number of query per second for all the queries in five different triplets datasets sizes, Table 6 and Table 7:**

		Number of performed queries per second									
		Q1a	Q1b	Q2a	Q2b	Q3a	Q3b	Q4a	Q4b	Q5a	Q5b
No. of triplets in set	10k	4484.4538	35.2259	4577.3706	26.7345	3722.7220	34.9394	8958.3586	1.4621	6101.1833	0.3547
	50k	4198.6616	8.3245	4277.1689	8.4989	4436.2689	8.6595	5864.4078	0.0274	5799.0586	0.0073
	150k	3549.0079	2.1819	3533.5356	1.6326	3645.3706	2.7039	6102.5870	0.0033	3116.9111	0.00078
	250k	1373.7150	1.0598	1322.6096	0.5980	1024.0026	0.8662	2777.8935	0.0022	2594.3432	0.00052
	500k	1383.1086	0.6508	1410.5486	0.3912	1061.1802	0.5242	2199.5521	0.00061	2483.7253	0.00024

Table 6 – Means for collected results for Queries Q1 to Q5 for all input data sizes

		Number of performed queries per second									
		Q6a	Q6b	Q7a	Q7b	Q8a	Q8b	Q9a	Q9b	Q10a	Q10b
No. of triplets in set	10k	4582.6708	237.5930	4686.9077	323.8678	1048.4379	300.4064	1239.9385	754.2522	897.6364	527.5763
	50k	4569.8371	72.8850	4565.6379	989.9032	1119.5005	989.9032	1147.2851	456.7742	1176.1265	896.3411
	150k	6575.7600	15.9429	5894.9699	25.6153	1170.7500	175.4067	1244.3399	47.1407	196.5601	83.5844
	250k	2757.5100	14.5715	2633.2975	26.8990	238.0476	166.5820	482.8678	25.6997	267.8611	23.9140
	500k	2673.4322	6.9348	2628.0380	25.5906	234.8807	166.3398	502.2266	21.4849	288.5449	21.0085

Table 7 – Means for collected results for Queries Q6 to Q10 for all input data sizes

4. Overview of performance of all queries over all triplets datasets, with focus on query (Fig. 1) and focus on dataset size (Fig. 2)

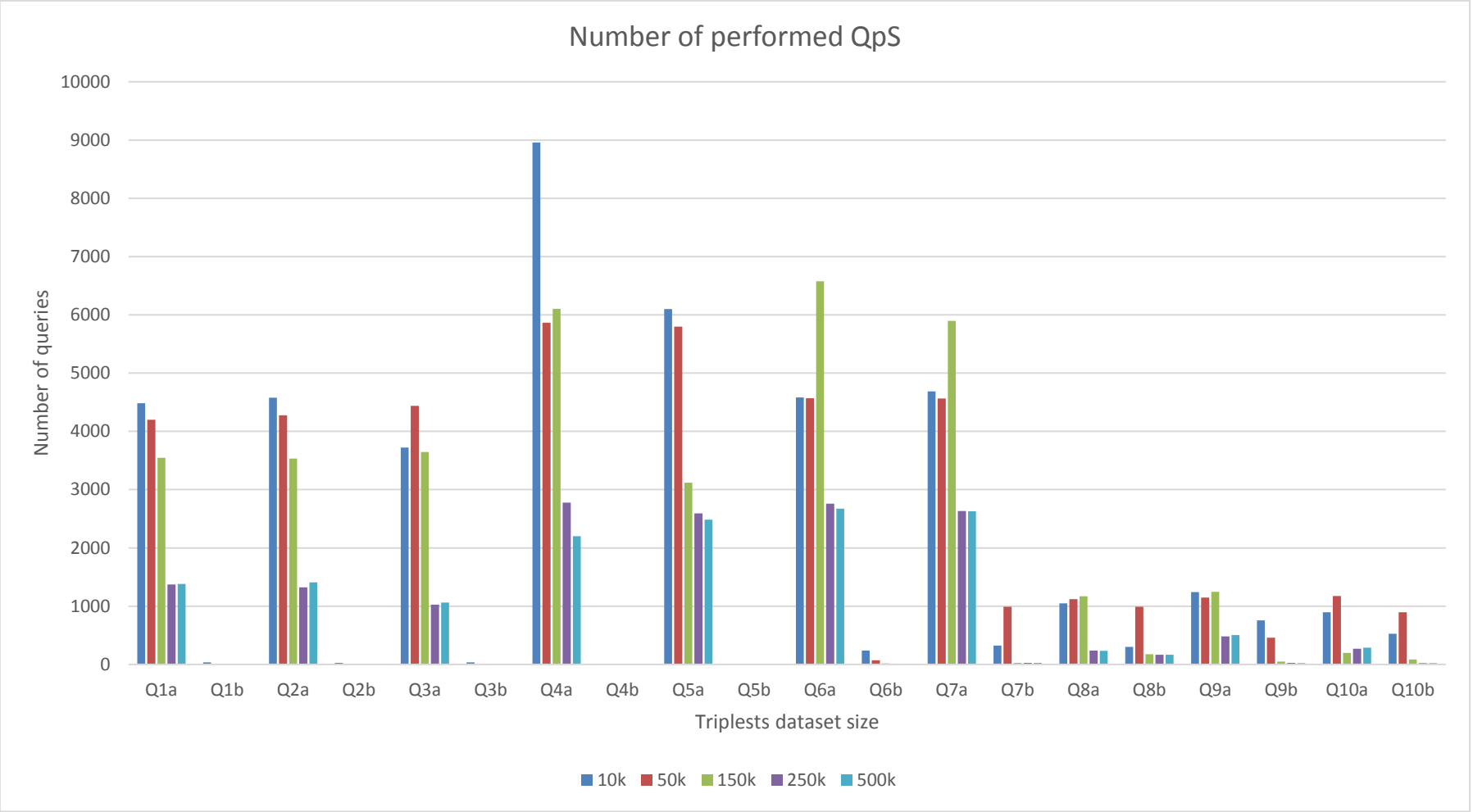


Figure 1 - Overall comparison between performed queries on QpS results

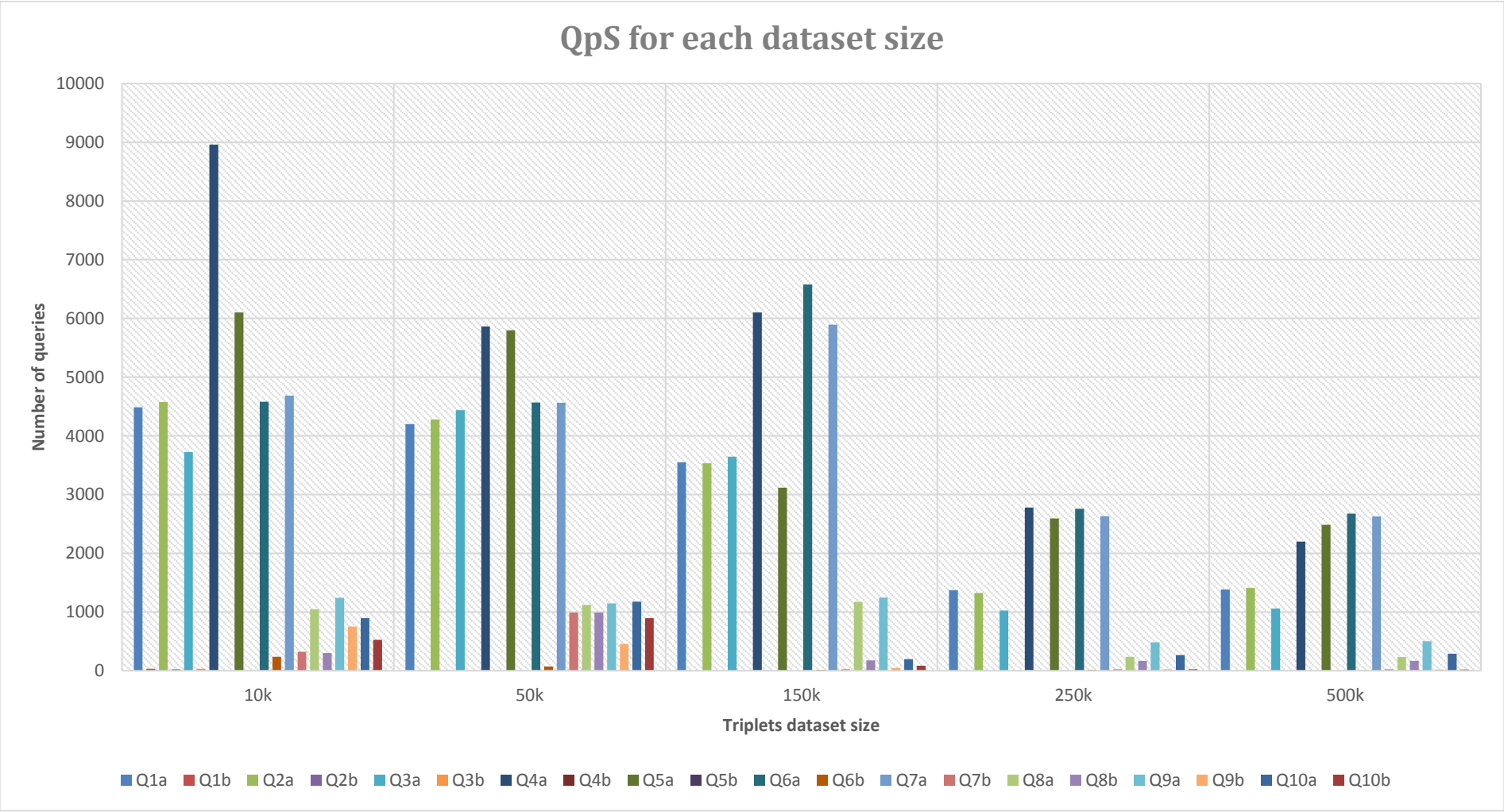
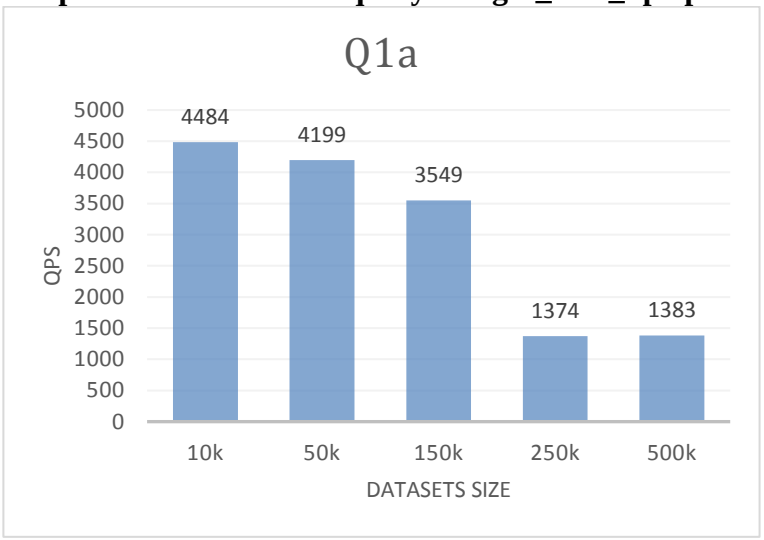
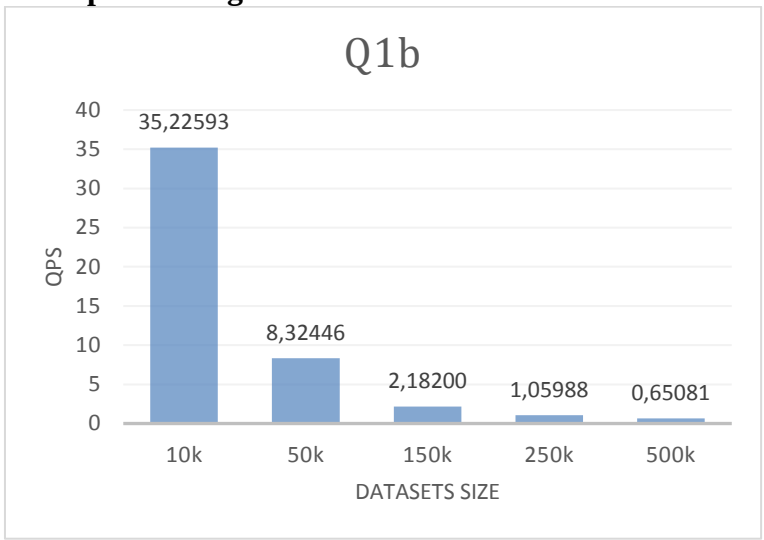
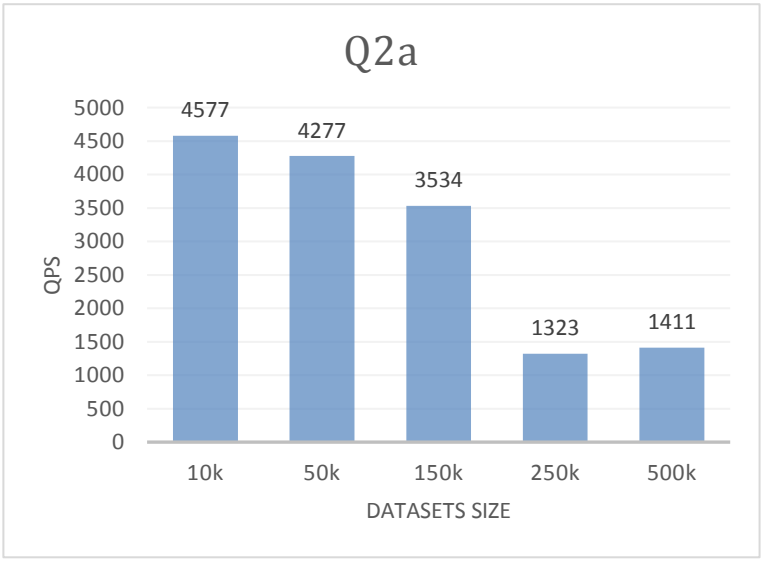


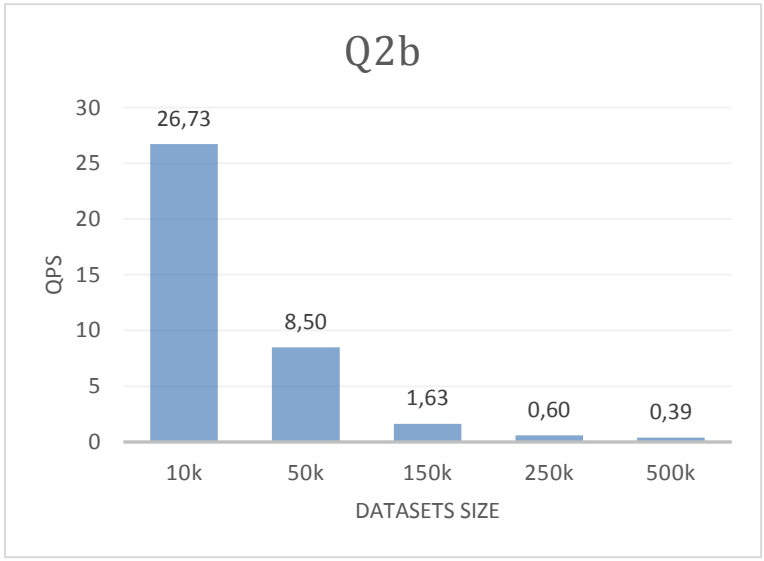
Figure 2 - Overall comparison between the performances of the queries based on the results per dataset size

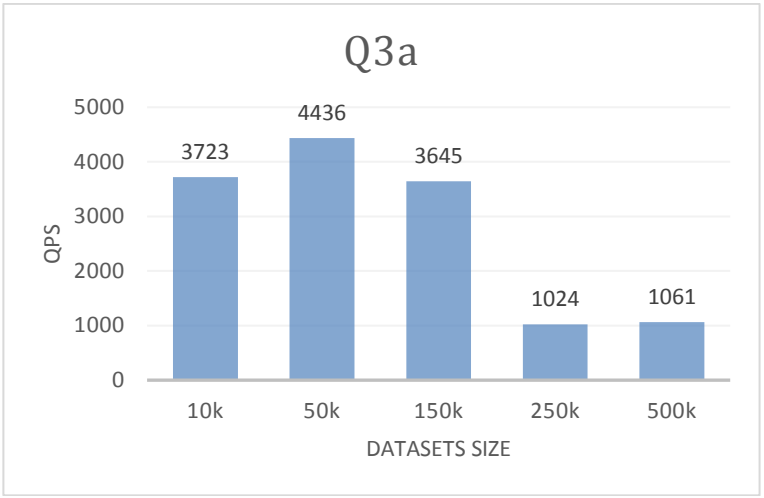
5. Visualised performance of each query using F_Roll_up operator and performing over all datasets

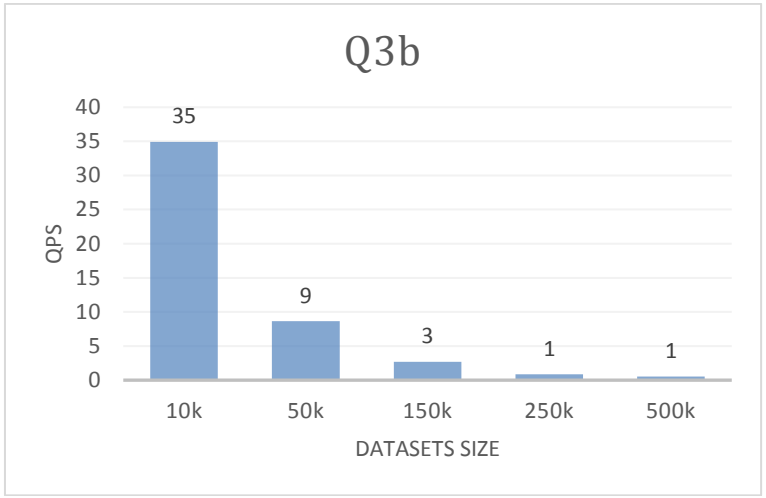









Q3a
5000
4000
3000
2000
1000
0
QPS
3723
4436
3645
1024
1061
10k
50k
150k
250k
500k
DATASETS SIZE


Q3b
40
35
30
25
20
15
10
5
0
QPS
35
9
3
1
1
10k
50k
150k
250k
500k
DATASETS SIZE

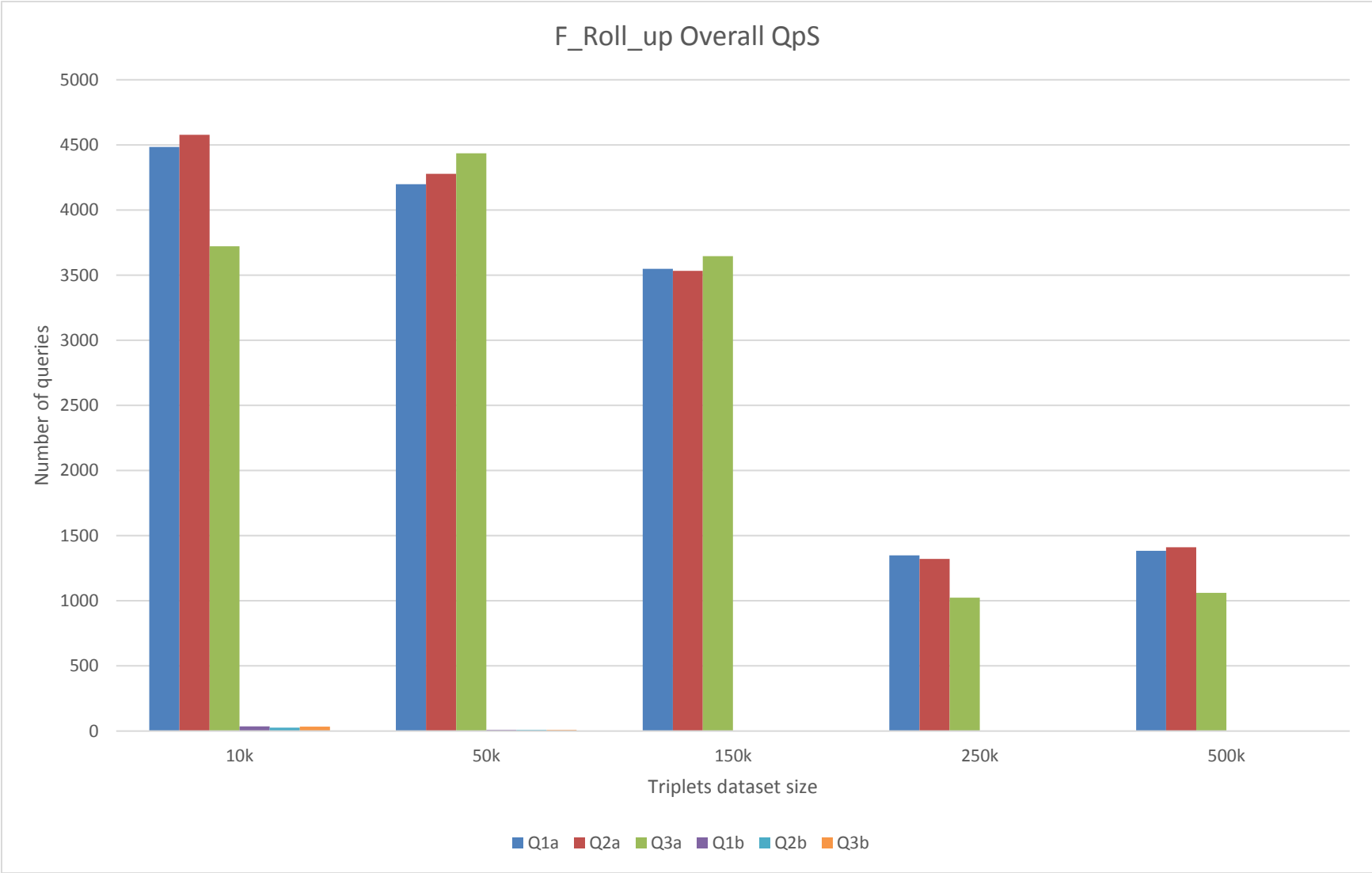
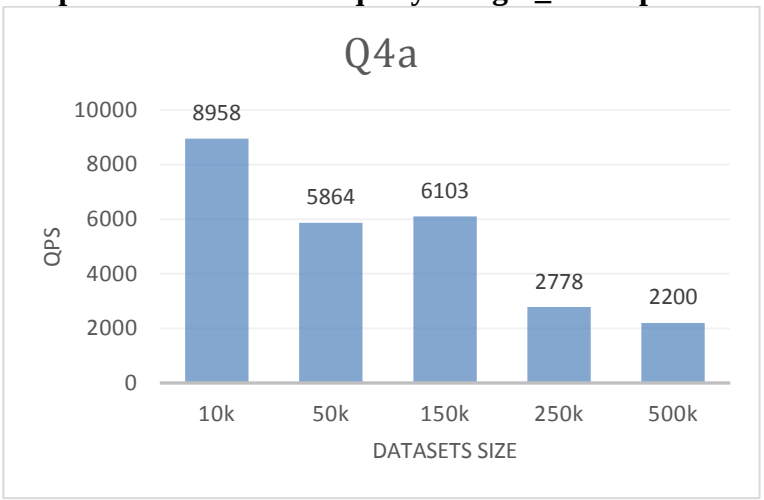
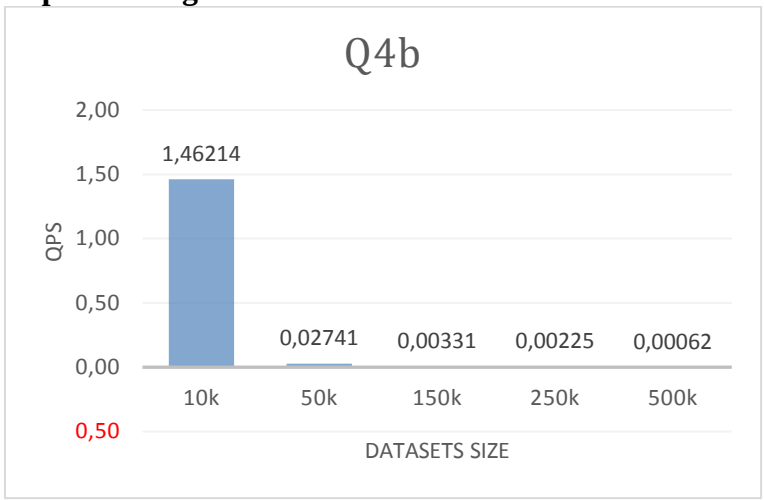
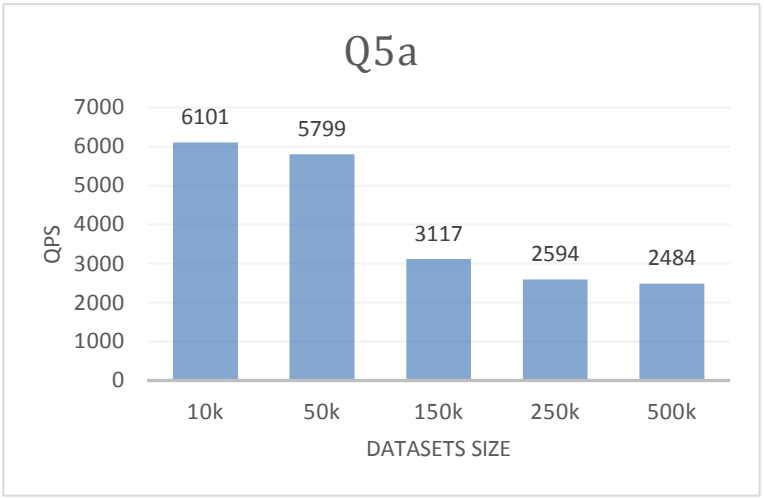


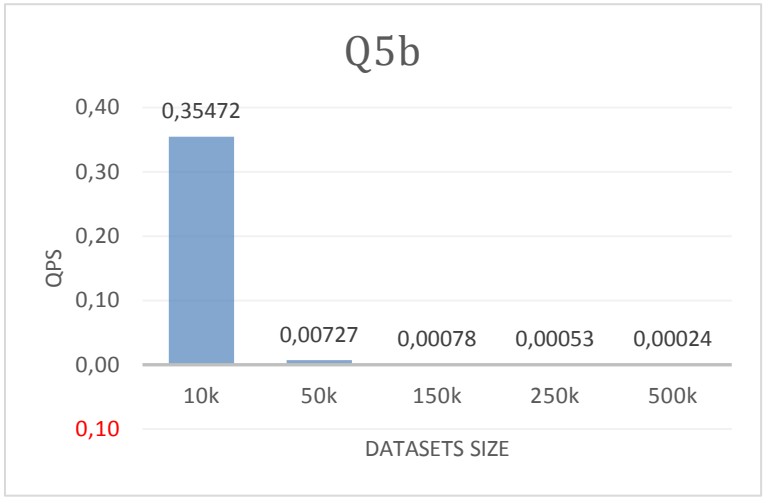
Figure 3 - Performance of all F_Roll_up queries

6. Visualised performance of each query using F_Drill operator and performing over all datasets









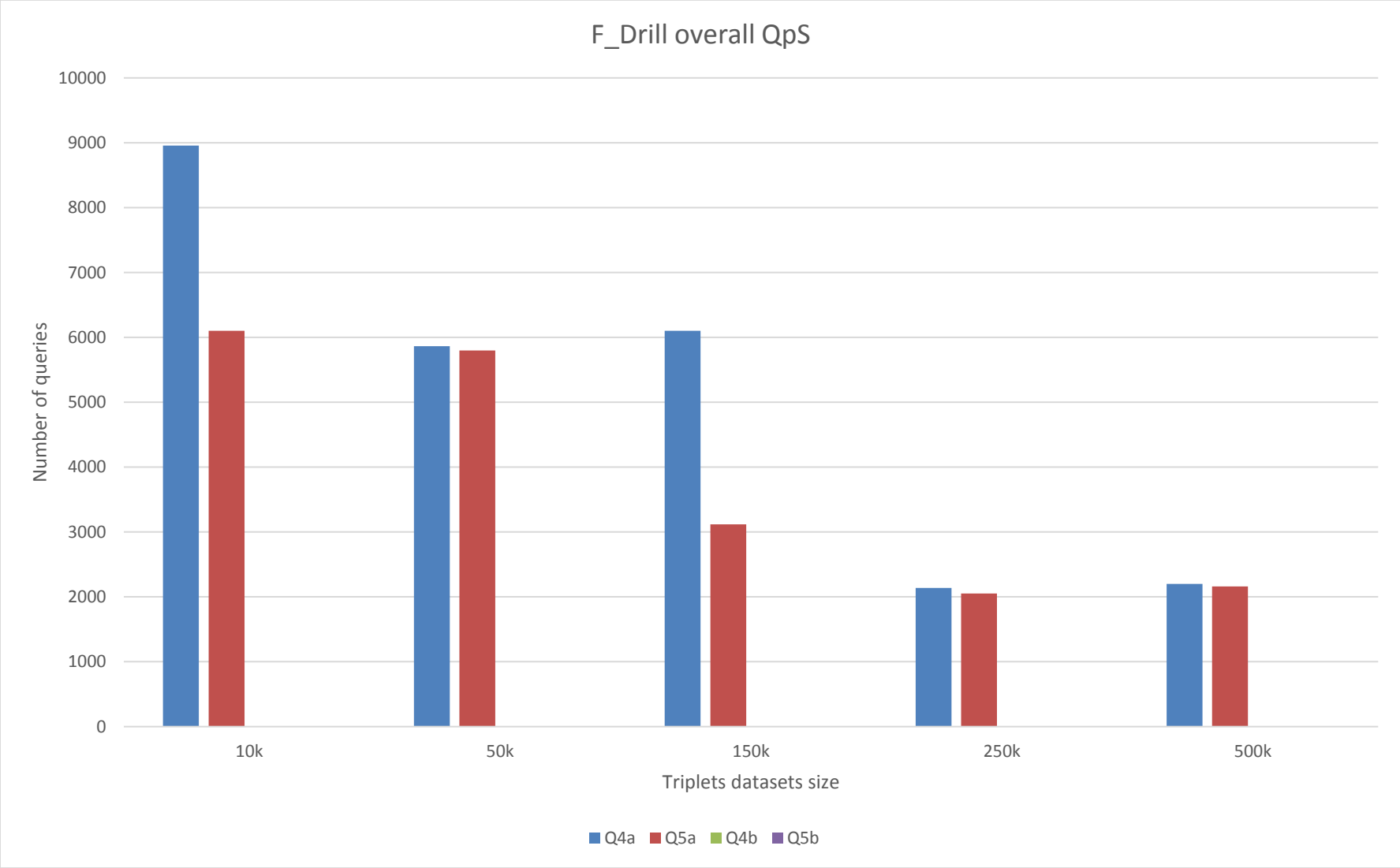
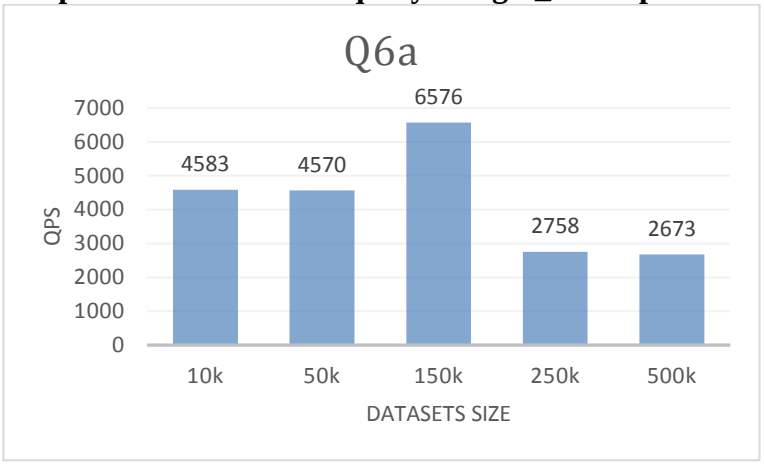
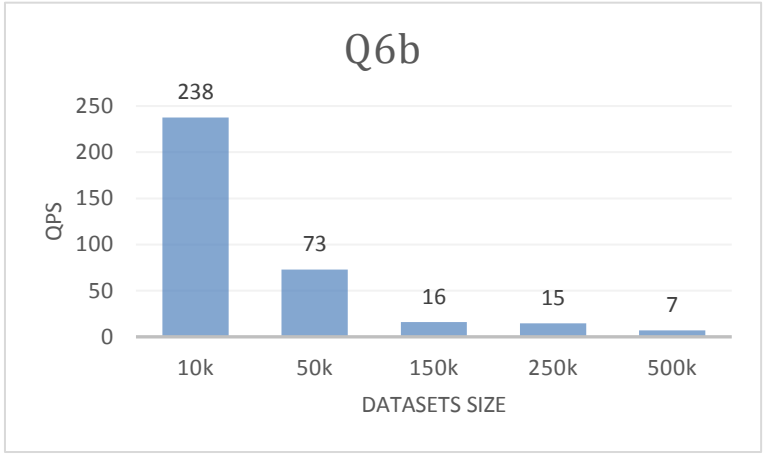
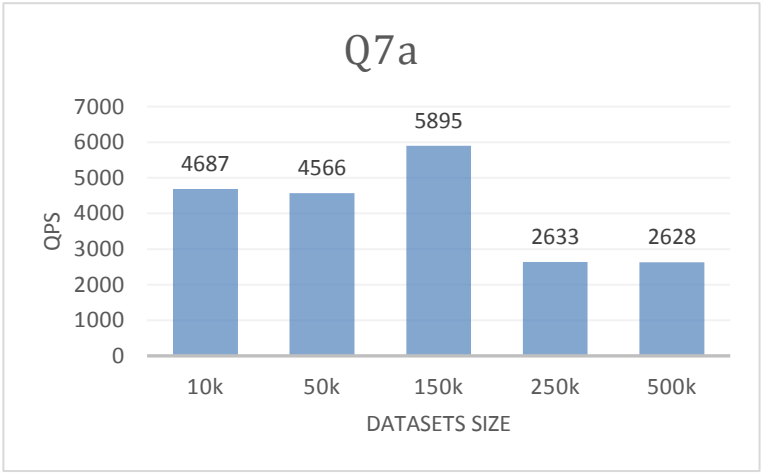


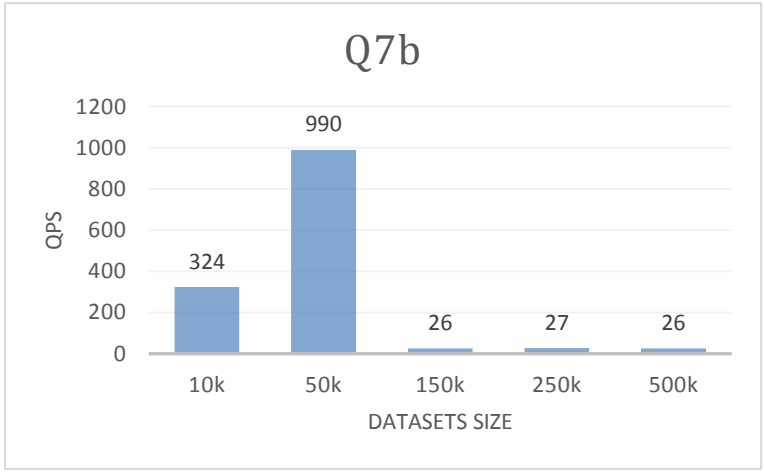
Figure 4 - Performance of all F_Drill queries

7. Visualised performance of each query using F_Slice operator and performing over all datasets









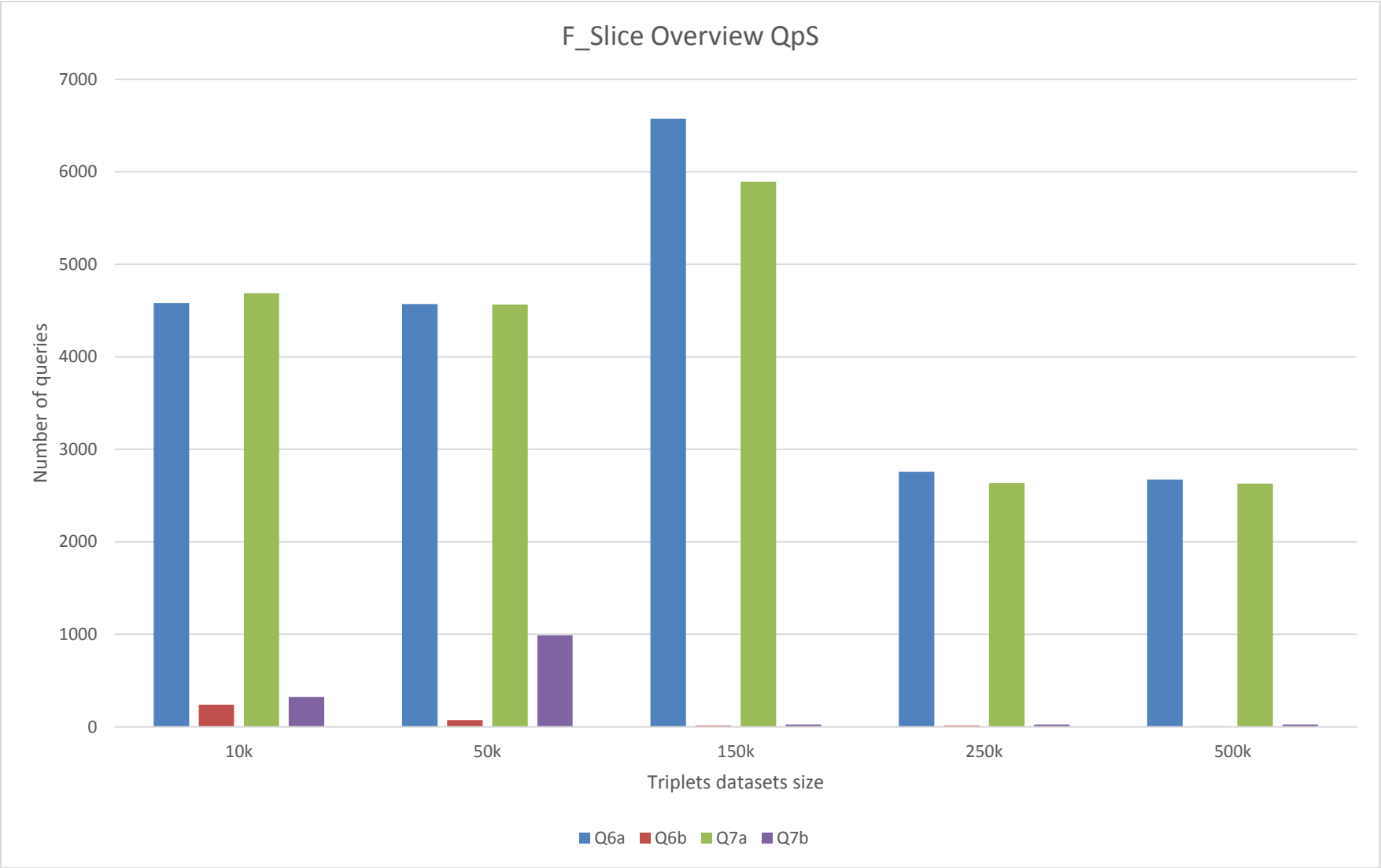
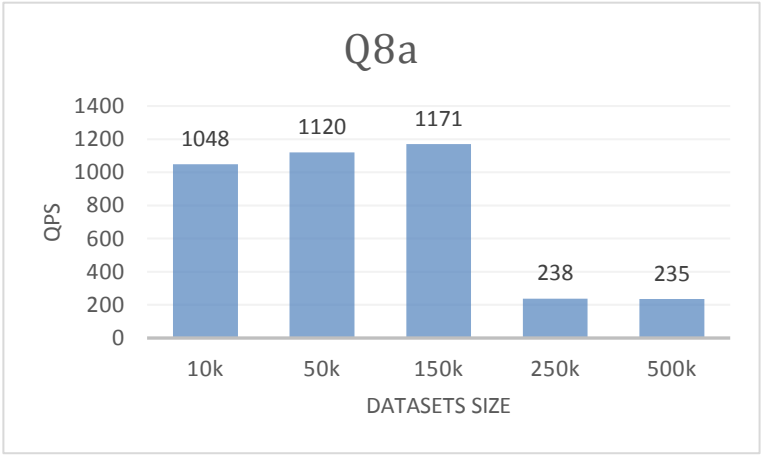
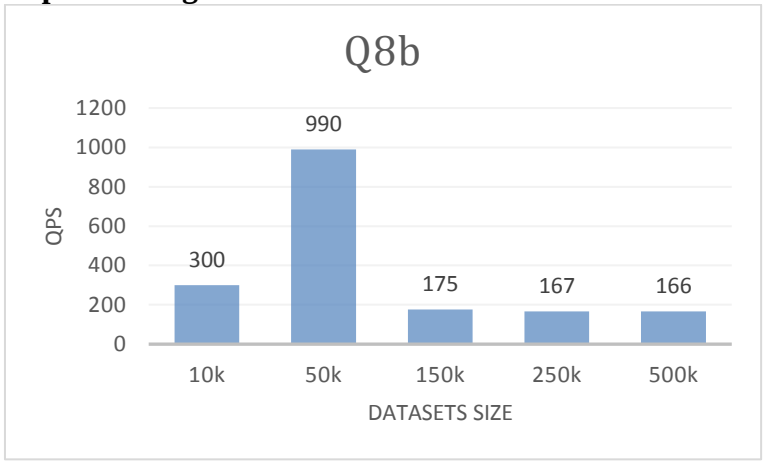
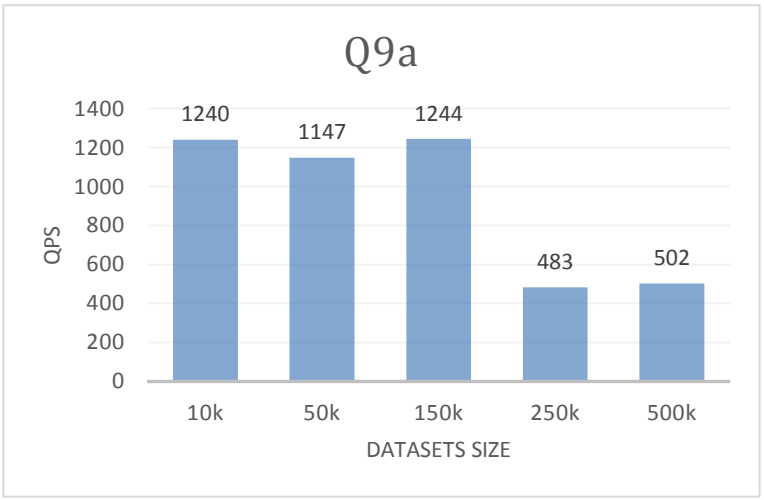


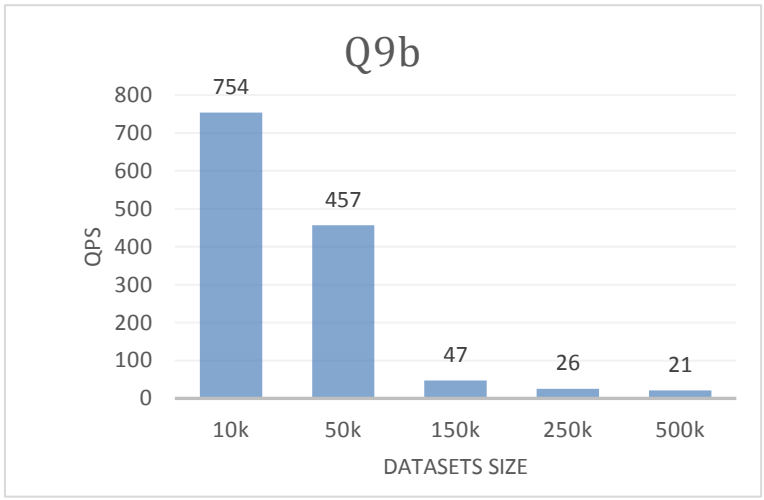
Figure 5 - Performance of all F_Slice queries

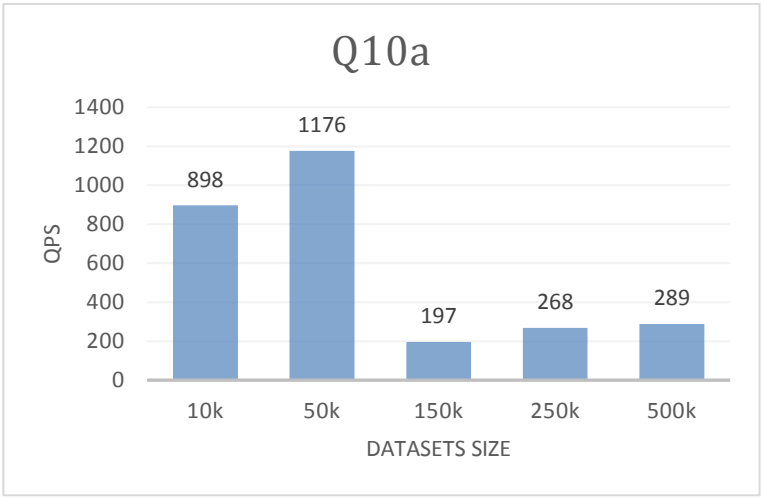
8. Visualised performance of each query using F_Slice operator and performing over all datasets

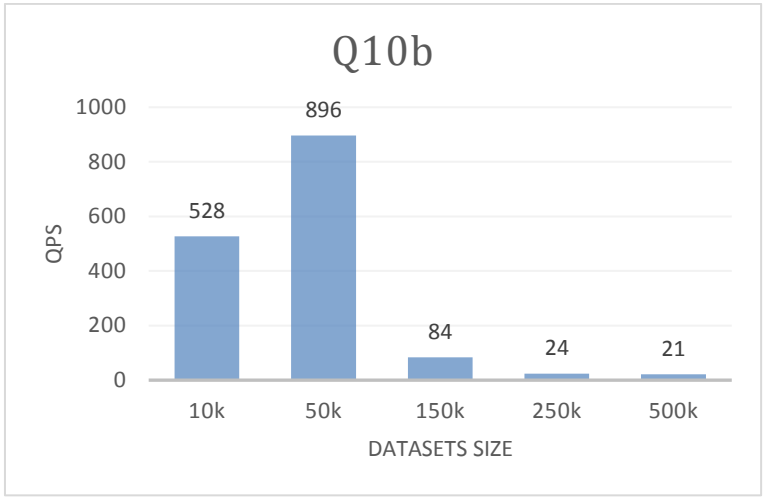









Q10a
1400
1200
1000
800
600
400
200
0
QPS
898
1176
197
268
289
10k
50k
150k
250k
500k
DATASETS SIZE


Q10b
1000
800
600
400
200
0
QPS
528
896
84
24
21
10k
50k
150k
250k
500k
DATASETS SIZE

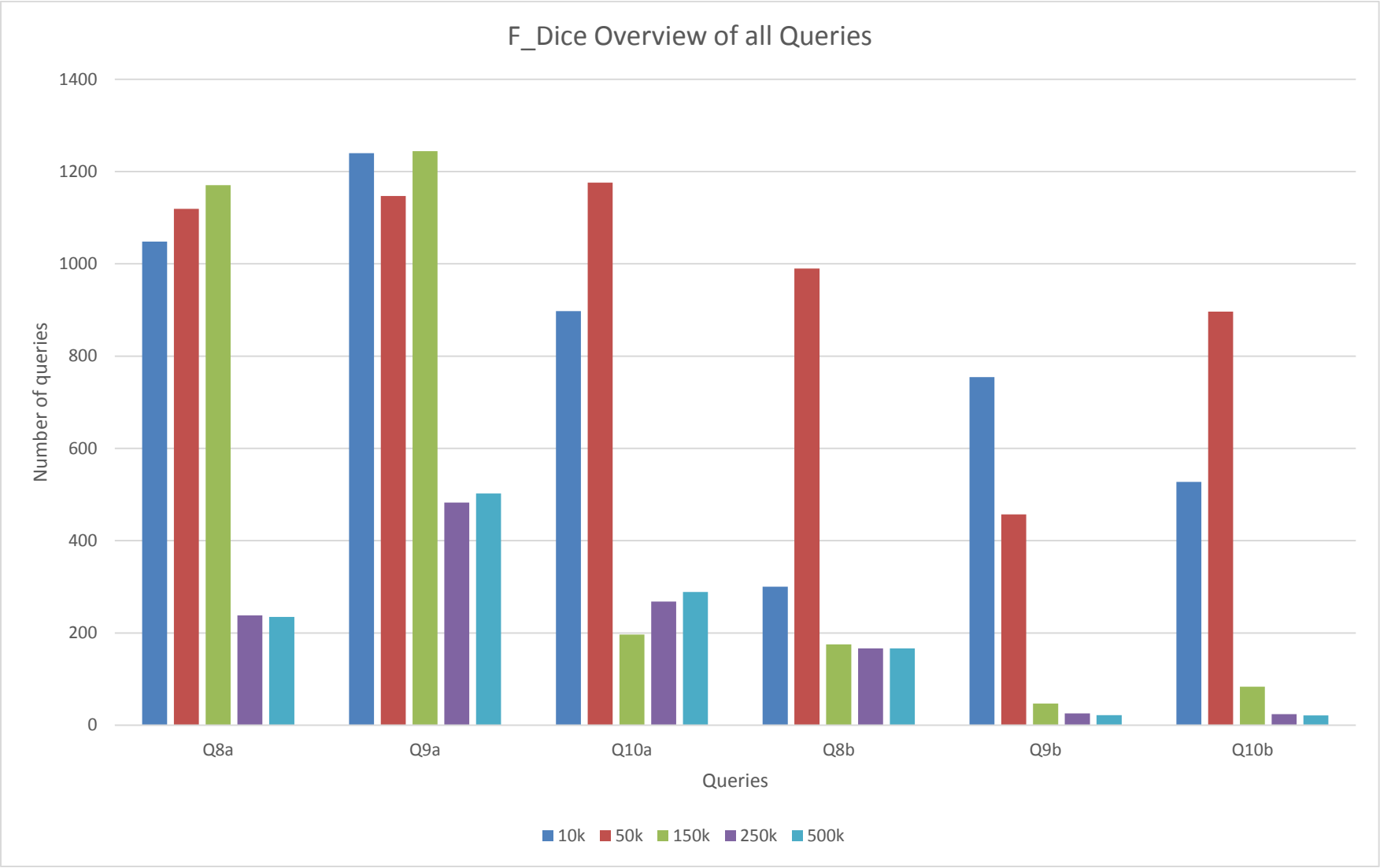


Figure 6 - Performance of all F_Dice queries